Red Hat 3scale API Management 2.4

Infrastructure

Learn more about deploying Red Hat 3scale API Management on different platforms.

Last Updated: 2021-09-13

Red Hat 3scale API Management 2.4 Infrastructure

Learn more about deploying Red Hat 3scale API Management on different platforms.

Legal Notice

Abstract

This guide documents deployment and infrastructure management with Red Hat 3scale API Management 2.4.

Table of Contents

CHAPTER 1. UPGRADING 3SCALE API MANAGEMENT 2.3 TO 2.4

As a 3scale API Management administrator, upgrade your installation from version 2.3 to 2.4. Optionally, you can change the impersonation of administrator data.

WARNING

This process can cause disruption in the service. Make sure to have a maintenance window.

1.1. PREREQUISITES

- 3scale API Management 2.3 deployed in a project.
- Tool prerequisites:
  - jq

1.2. UPGRADING 3SCALE API MANAGEMENT

To upgrade 3scale API Management from 2.3 to 2.4 follow the steps in the order listed below:

1. Section 1.2.1, “Creating ConfigMaps”
2. Section 1.2.2, “Creating the system master route”
3. Section 1.2.3, “Migrating the **system** database secret”
4. Section 1.2.4.2, “Creating the **-redis** secrets”
5. Section 1.2.5, “Patching DeploymentConfigs”
6. Section 1.2.6, “Patching image streams”
7. Section 1.2.7, “Adding the entry for Service Discovery”
8. Section 1.2.8, “Configuring additional DeploymentConfigs”

1.2.1. Creating ConfigMaps

Create files that contain the details of the ConfigMaps to be sourced for the new OpenShift elements.

1. Configure the required variables:

```
export $(oc set env dc/system-app --list|grep THREESCALE_SUPERDOMAIN|sort -u)
export $(oc set env dc/system-app --list|grep APICAST_REGISTRY_URL|sort -u)
APP_LABEL=$(oc get dc backend-listener -o json | jq .spec.template.metadata.labels.app -r)
AMP_RELEASE=2.4.0
```

2. Confirm that the variables are properly configured:

```
echo $THREESCALE_SUPERDOMAIN
echo $APICAST_REGISTRY_URL
echo $APP_LABEL
echo $AMP_RELEASE
```

3. Create the **system-environment** ConfigMap:

   a. Compose a file called **system-environment.yml**.

```
cat<<EOF> system-environment.yml

apiVersion: v1
data:
  AMP_RELEASE: ${AMP_RELEASE}
  APICAST_REGISTRY_URL: ${APICAST_REGISTRY_URL}
  FORCE_SSL: "true"
  PROVIDER_PLAN: enterprise
  RAILS_ENV: production
  RAILS_LOG_LEVEL: info
  RAILS_LOG_TO_STDOUT: "true"
  SSL_CERT_DIR: /etc/pki/tls/certs
  THINKING_SPHINX_PORT: "9306"
  THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE: VERIFY_NONE
  THREESCALE_SUPERDOMAIN: ${THREESCALE_SUPERDOMAIN}
kind: ConfigMap
metadata:
  creationTimestamp: null
  labels:
    app: ${APP_LABEL}
    3scale.component: system
  name: system-environment
EOF
```

   b. Create the new **system-environment** ConfigMap:

```
oc create -f system-environment.yml
```

4. Create the **backend-environment** ConfigMap:

   a. Get the value of the **backend-listener** DeploymentConfig:

```
export $(oc set env dc/backend-listener --list|grep RACK_ENV|sort -u)
```

   b. Compose a new file called **backend-environment.yml**.

```
cat<<EOF>backend-environment.yml
apiVersion: v1
data:
  RACK_ENV: ${RACK_ENV}
kind: ConfigMap
metadata:
  creationTimestamp: null
  labels:
```

```
    app: ${APP_LABEL}
    3scale.component: backend
  name: backend-environment
EOF
```

c. Create the new **backend-environment** ConfigMap:

```
oc create -f backend-environment.yml
```

5. Create the **apicast-environment** ConfigMap:

a. Compose a new file called **apicast-environment.yml**.

```
cat<<EOT>apicast-environment.yml
apiVersion: v1
data:
  APICAST_MANAGEMENT_API: status
  APICAST_RESPONSE_CODES: "true"
  OPENSSL_VERIFY: "false"
kind: ConfigMap
metadata:
  name: apicast-environment
EOT
```

b. Create the new **apicast-environment** ConfigMap:

```
oc create -f apicast-environment.yml
```

1.2.2. Creating the system master route

To create the system master route:

1. Configure **MASTER_NAME** with the value of the **MASTER_DOMAIN** environment variable:

```
export $(oc set env dc/system-app --list|grep MASTER_DOMAIN|sort -u)
MASTER_NAME=$MASTER_DOMAIN
```

2. Confirm that **MASTER_NAME** is properly set:

```
echo $MASTER_NAME
```

3. Create the **system-master** route and delete the **system-master-admin** route:

```
oc create route edge system-master --service=system-master --
hostname=${MASTER_NAME}.${THREESCALE_SUPERDOMAIN} --port=http
oc delete route system-master-admin
```

1.2.3. Migrating the system database secret

To migrate the **system** database secret into the new **system-database** OpenShift secret:

1. Get the existing MySQL environment variables:

```
export $(oc set env dc/system-mysql --list|grep MYSQL_ROOT_PASSWORD)
export $(oc set env dc/system-mysql --list | grep MYSQL_DATABASE)
```

2. Get the APP_LABEL environment variable:

```
APP_LABEL=$(oc get dc backend-listener -o json | jq .spec.template.metadata.labels.app -r)
```

3. Confirm that you have correctly set the following environment variables:

```
echo ${MYSQL_ROOT_PASSWORD}
echo ${MYSQL_DATABASE}
echo ${APP_LABEL}
```

4. Create the file containing the **system-database** secret:

```
cat > system-database.yml <<EOF

apiVersion: v1
kind: Secret
metadata:
  creationTimestamp: null
  labels:
    3scale.component: system
    app: ${APP_LABEL}
  name: system-database
stringData:
  URL: mysql2://root:${MYSQL_ROOT_PASSWORD}@system-
mysql/${MYSQL_DATABASE}
type: Opaque
EOF
```

5. Create the secret:

```
oc create -f system-database.yml
```

1.2.4. Creating new secrets

To continue with the upgrade process, you must create:

- The **system-master-apicast** secret
- Several **-redis** secrets
- Several **backend-** secrets
- Several **system-** secrets

1.2.4.1. Creating the system-master-apicast secret

1. Get the value of APICAST_ACCESS_TOKEN:

```
export APICAST_ACCESS_TOKEN=$(oc set env --list dc/apicast-production|sort -u|grep
THREESCALE_PORTAL_ENDPOINT|cut -d@ -f1|cut -d/ -f3)
```

2. Compose a new file called **system-master-apicast.yml**:

```
cat<<EOF> system-master-apicast.yml
apiVersion: v1
kind: Secret
metadata:
  creationTimestamp: null
  labels:
    app: ${APP_LABEL}
    3scale.component: system
  name: system-master-apicast
stringData:
  ACCESS_TOKEN: ${APICAST_ACCESS_TOKEN}
  BASE_URL: http://${APICAST_ACCESS_TOKEN}@system-master:3000
  PROXY_CONFIGS_ENDPOINT: http://${APICAST_ACCESS_TOKEN}@system-
master:3000/master/api/proxy/configs
type: Opaque
EOF
```

3. Create the new **system-master-apicast** secret:

```
oc create -f system-master-apicast.yml
```

Back to Section 1.2.4, “Creating new secrets” .

1.2.4.2. Creating the -redis secrets

1. Create the **system-redis** secret:

   a. Compose a file called **system-redis.yml** with the following content:

```
cat > system-redis.yml <<EOF

apiVersion: v1
kind: Secret
metadata:
  creationTimestamp: null
  labels:
     3scale.component: system
     app: ${APP_LABEL}
  name: system-redis
stringData:
  URL: redis://system-redis:6379/1
  type: Opaque
EOF
```

   b. Create the **system-redis** secret with the information contained in the file:

```
oc create -f system-redis.yml
```

2. Create the **backend-redis** secret:

   a. Compose a file called **backend-redis.yml** with this content:

```
cat > backend-redis.yml <<EOF

apiVersion: v1
kind: Secret
type: Opaque
metadata:
  creationTimestamp: null
  labels:
     3scale.component: backend
     app: ${APP_LABEL}
  name: backend-redis
stringData:
  REDIS_QUEUES_SENTINEL_HOSTS: ""
  REDIS_QUEUES_SENTINEL_ROLE: ""
  REDIS_QUEUES_URL: redis://backend-redis:6379/1
  REDIS_STORAGE_SENTINEL_HOSTS: ""
  REDIS_STORAGE_SENTINEL_ROLE: ""
  REDIS_STORAGE_URL: redis://backend-redis:6379/0
EOF
```

b. Create the **backend-redis** secret with the information contained in the file:

```
oc create -f backend-redis.yml
```

3. Create the **apicast-redis** secret:

a. Compose a file called **apicast-redis.yml** with this content:

```
cat<<EOT>apicast-redis.yml
apiVersion: v1
stringData:
  PRODUCTION_URL: redis://system-redis:6379/1
  STAGING_URL: redis://system-redis:6379/2
kind: Secret
metadata:
  labels:
    3scale.component: apicast
    app: ${APP_LABEL}
  name: apicast-redis
EOT
```

b. Create the **apicast-redis** secret with the information contained in the file:

```
oc create -f apicast-redis.yml
```

Back to Section 1.2.4, “Creating new secrets” .

1.2.4.3. Creating the backend- secrets

1. Create the **backend-listener** secret:

a. Compose a file called **backend-listener.yml** with this content:

```
cat<<EOT>backend-listener.yml
```

```
apiVersion: v1
stringData:
  route_endpoint: https://backend-3scale.${THREESCALE_SUPERDOMAIN}
  service_endpoint: http://backend-listener:3000
kind: Secret
metadata:
  name: backend-listener
EOT
```

b. Create the **backend-listener** secret with the information contained in the file:

```
oc create -f backend-listener.yml
```

2. Create the **backend-internal-api** secret:

a. Obtain the values from the 2.3 environment:

```
export $(oc set env dc backend-listener --list|grep CONFIG_INTERNAL_API_USER)
export $(oc set env dc backend-listener --list|grep
CONFIG_INTERNAL_API_PASSWORD)
```

b. Check the value of the variables:

```
echo $CONFIG_INTERNAL_API_USER $CONFIG_INTERNAL_API_PASSWORD
```

c. Create the secret:

```
oc create secret generic backend-internal-api --from-
literal=password=${CONFIG_INTERNAL_API_PASSWORD} --from-
literal=username=${CONFIG_INTERNAL_API_USER}
```

Back to Section 1.2.4, “Creating new secrets” .

1.2.4.4. Creating the system- secrets

1. Create the **system-memcache** secret:

a. Compose a file called **system-memcache.yml** with this content:

```
cat<<EOT>system-memcache.yml
apiVersion: v1
stringData:
  SERVERS: system-memcache:11211
kind: Secret
metadata:
  name: system-memcache
EOT
```

b. Create the **system-memcache** secret with the information contained in the file:

```
oc create -f system-memcache.yml
```

2. Create the **system-recaptcha** secret:

a. Compose a file called **system-recaptcha.yml** with this content:

```
cat<<EOT>system-recaptcha.yml
apiVersion: v1
stringData:
  PRIVATE_KEY: ""
  PUBLIC_KEY: ""
kind: Secret
metadata:
  name: system-recaptcha
EOT
```

b. Create the **system-recaptcha** secret with the information contained in the file:

```
oc create -f system-recaptcha.yml
```

3. Create the **system-events-hook** secret:

a. Get the values from the 2.3 environment:

```
export $(oc set env dc backend-worker --list|grep
CONFIG_EVENTS_HOOK_SHARED_SECRET)
```

b. Confirm the value of the variables:

```
echo ${CONFIG_EVENTS_HOOK_SHARED_SECRET}
```

c. Create the secret:

```
oc create secret generic system-events-hook --from-
literal=PASSWORD=${CONFIG_EVENTS_HOOK_SHARED_SECRET} --from-
literal=URL=http://system-master:3000/master/events/import
```

4. Create the **system-seed** secret:

a. Get the values from the 2.3 environment:

```
export $(oc set env dc system-app --list|grep ADMIN_ACCESS_TOKEN|uniq)
export $(oc set env dc system-app --list|grep MASTER_ACCESS_TOKEN|uniq)
export $(oc set env dc system-app --list|grep USER_PASSWORD|uniq)
export $(oc set env dc system-app --list|grep USER_LOGIN|uniq)
export $(oc set env dc system-app --list|grep MASTER_DOMAIN|uniq)
export $(oc set env dc system-app --list|grep MASTER_PASSWORD|uniq)
export $(oc set env dc system-app --list|grep MASTER_USER|uniq)
export $(oc set env dc system-app --list|grep TENANT_NAME|uniq)
```

b. Confirm the value of the variables:

```
echo ${ADMIN_ACCESS_TOKEN} ${MASTER_ACCESS_TOKEN}
${USER_PASSWORD} ${USER_LOGIN} ${MASTER_DOMAIN}
${MASTER_PASSWORD} ${MASTER_USER} ${TENANT_NAME}
```

c. Create the secret:

```
oc create secret generic system-seed --from-
literal=ADMIN_ACCESS_TOKEN=${ADMIN_ACCESS_TOKEN} --from-
literal=MASTER_ACCESS_TOKEN=${MASTER_ACCESS_TOKEN} --from-
literal=ADMIN_PASSWORD=${USER_PASSWORD} --from-
literal=ADMIN_USER=${USER_LOGIN} --from-
literal=MASTER_DOMAIN=${MASTER_DOMAIN} --from-
literal=MASTER_PASSWORD=${MASTER_PASSWORD} --from-
literal=MASTER_USER=${MASTER_USER} --from-
literal=TENANT_NAME=${TENANT_NAME}
```

5. Create the **system-app** secret:

   a. Obtain the values from the 2.3 environment:

   ```
   export $(oc set env dc system-app --list|grep SECRET_KEY_BASE|uniq)
   ```

   b. Confirm the value of the variables:

   ```
   echo ${SECRET_KEY_BASE}
   ```

   c. Create the secret:

   ```
   oc create secret generic system-app --from-
   literal=SECRET_KEY_BASE=${SECRET_KEY_BASE}
   ```

Back to Section 1.2.4, “Creating new secrets” .

1.2.5. Patching DeploymentConfigs

1. Patch the **backend-cron** DeploymentConfig:

```
oc patch dc/backend-cron -p '{"spec":{"template":{"spec":{"containers":[{"name":"backend-
cron","env":[{"name":"CONFIG_REDIS_PROXY","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_URL","name":"backend-redis"}}},
{"name":"CONFIG_REDIS_SENTINEL_HOSTS","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_SENTINEL_HOSTS","name":"backend-redis"}}},
{"name":"CONFIG_REDIS_SENTINEL_ROLE","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_SENTINEL_ROLE","name":"backend-redis"}}},
{"name":"CONFIG_QUEUES_MASTER_NAME","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_URL","name":"backend-redis"}}},
{"name":"CONFIG_QUEUES_SENTINEL_HOSTS","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_SENTINEL_HOSTS","name":"backend-redis"}}},
{"name":"CONFIG_QUEUES_SENTINEL_ROLE","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_SENTINEL_ROLE","name":"backend-redis"}}},
{"name":"RACK_ENV","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RACK_ENV","name":"backend-environment"}}}]}],"initContainers":[{"name":"backend-
redis-svc","command":["/opt/app/entrypoint.sh","sh","-c","until rake
connectivity:redis_storage_queue_check; do sleep $SLEEP_SECONDS;\ndone"],"env":
[{"name":"CONFIG_QUEUES_MASTER_NAME","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_URL","name":"backend-redis"}}}]}]}}}}'
```

2. Patch the **backend-listener** DeploymentConfig:

```
oc patch dc/backend-listener -p '{"spec":{"template":{"spec":{"containers":[{"name":"backend-
```

```
listener","env":[{"name":"CONFIG_REDIS_PROXY","value":null,"valueFrom":
{"secretKeyRef":{"key":"REDIS_STORAGE_URL","name":"backend-redis"}}},
{"name":"CONFIG_REDIS_SENTINEL_HOSTS","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_SENTINEL_HOSTS","name":"backend-redis"}}},
{"name":"CONFIG_REDIS_SENTINEL_ROLE","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_SENTINEL_ROLE","name":"backend-redis"}}},
{"name":"CONFIG_QUEUES_MASTER_NAME","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_URL","name":"backend-redis"}}},
{"name":"CONFIG_QUEUES_SENTINEL_HOSTS","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_SENTINEL_HOSTS","name":"backend-redis"}}},
{"name":"CONFIG_QUEUES_SENTINEL_ROLE","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_SENTINEL_ROLE","name":"backend-redis"}}},
{"name":"RACK_ENV","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RACK_ENV","name":"backend-environment"}}},
{"name":"CONFIG_INTERNAL_API_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"username","name":"backend-internal-api"}}},
{"name":"CONFIG_INTERNAL_API_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"password","name":"backend-internal-api"}}}]}}}}'
```

3. Patch the **backend-worker** DeploymentConfig:

```
oc patch dc/backend-worker -p '{"spec":{"template":{"spec":{"containers":[{"name":"backend-
worker","env":[{"name":"CONFIG_REDIS_PROXY","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_URL","name":"backend-redis"}}},
{"name":"CONFIG_REDIS_SENTINEL_HOSTS","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_SENTINEL_HOSTS","name":"backend-redis"}}},
{"name":"CONFIG_REDIS_SENTINEL_ROLE","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_SENTINEL_ROLE","name":"backend-redis"}}},
{"name":"CONFIG_QUEUES_MASTER_NAME","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_URL","name":"backend-redis"}}},
{"name":"CONFIG_QUEUES_SENTINEL_HOSTS","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_SENTINEL_HOSTS","name":"backend-redis"}}},
{"name":"CONFIG_QUEUES_SENTINEL_ROLE","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_SENTINEL_ROLE","name":"backend-redis"}}},
{"name":"RACK_ENV","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RACK_ENV","name":"backend-environment"}}},
{"name":"CONFIG_EVENTS_HOOK","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-events-hook"}}},
{"name":"CONFIG_EVENTS_HOOK_SHARED_SECRET","value":null,"valueFrom":
{"secretKeyRef":{"key":"PASSWORD","name":"system-events-hook"}}}]}],"initContainers":
[{"name":"backend-redis-svc","command":["/opt/app/entrypoint.sh","sh","-c","until rake
connectivity:redis_storage_queue_check; do sleep $SLEEP_SECONDS;\ndone"],"env":
[{"name":"CONFIG_QUEUES_MASTER_NAME","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_QUEUES_URL","name":"backend-redis"}}}]}]}}}}'
```

4. Patch the **system-app** DeploymentConfig containers:

```
oc patch dc/system-app -p '{"spec":{"strategy":{"activeDeadlineSeconds":21600,"resources":
{},"rollingParams":{"pre":{"execNewPod":{"env":
[{"name":"AMP_RELEASE","value":null,"valueFrom":{"configMapKeyRef":
{"key":"AMP_RELEASE","name":"system-environment"}}},
{"name":"APICAST_REGISTRY_URL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"APICAST_REGISTRY_URL","name":"system-environment"}}},
{"name":"FORCE_SSL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"FORCE_SSL","name":"system-environment"}}},
```

{"name":"PROVIDER_PLAN","value":null,"valueFrom":{"configMapKeyRef":
{"key":"PROVIDER_PLAN","name":"system-environment"}}},
{"name":"RAILS_ENV","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_ENV","name":"system-environment"}}},
{"name":"RAILS_LOG_LEVEL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_LOG_LEVEL","name":"system-environment"}}},
{"name":"RAILS_LOG_TO_STDOUT","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_LOG_TO_STDOUT","name":"system-environment"}}},
{"name":"SSL_CERT_DIR","value":null,"valueFrom":{"configMapKeyRef":
{"key":"SSL_CERT_DIR","name":"system-environment"}}},
{"name":"THINKING_SPHINX_PORT","value":null,"valueFrom":{"configMapKeyRef":
{"key":"THINKING_SPHINX_PORT","name":"system-environment"}}},
{"name":"THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE","value":null,"value
From":{"configMapKeyRef":
{"key":"THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE","name":"system-
environment"}}},{"name":"THREESCALE_SUPERDOMAIN","value":null,"valueFrom":
{"configMapKeyRef":{"key":"THREESCALE_SUPERDOMAIN","name":"system-
environment"}}},{"name":"DATABASE_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-database"}}},
{"name":"MASTER_DOMAIN","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_DOMAIN","name":"system-seed"}}},
{"name":"MASTER_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_USER","name":"system-seed"}}},
{"name":"MASTER_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_PASSWORD","name":"system-seed"}}},
{"name":"ADMIN_ACCESS_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_ACCESS_TOKEN","name":"system-seed"}}},
{"name":"USER_LOGIN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_USER","name":"system-seed"}}},
{"name":"USER_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_PASSWORD","name":"system-seed"}}},
{"name":"TENANT_NAME","value":null,"valueFrom":{"secretKeyRef":
{"key":"TENANT_NAME","name":"system-seed"}}},
{"name":"THINKING_SPHINX_ADDRESS","value":"system-sphinx"},
{"name":"THINKING_SPHINX_CONFIGURATION_FILE","value":"/tmp/sphinx.conf"},
{"name":"EVENTS_SHARED_SECRET","value":null,"valueFrom":{"secretKeyRef":
{"key":"PASSWORD","name":"system-events-hook"}}},
{"name":"RECAPTCHA_PUBLIC_KEY","value":null,"valueFrom":{"secretKeyRef":
{"key":"PUBLIC_KEY","name":"system-recaptcha"}}},
{"name":"RECAPTCHA_PRIVATE_KEY","value":null,"valueFrom":{"secretKeyRef":
{"key":"PRIVATE_KEY","name":"system-recaptcha"}}},
{"name":"SECRET_KEY_BASE","value":null,"valueFrom":{"secretKeyRef":
{"key":"SECRET_KEY_BASE","name":"system-app"}}},
{"name":"REDIS_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-redis"}}},
{"name":"MEMCACHE_SERVERS","value":null,"valueFrom":{"secretKeyRef":
{"key":"SERVERS","name":"system-memcache"}}},
{"name":"BACKEND_REDIS_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_URL","name":"backend-redis"}}},
{"name":"APICAST_BACKEND_ROOT_ENDPOINT","value":null,"valueFrom":
{"secretKeyRef":{"key":"route_endpoint","name":"backend-listener"}}},
{"name":"BACKEND_ROUTE","value":null,"valueFrom":{"secretKeyRef":
{"key":"route_endpoint","name":"backend-listener"}}},
{"name":"SMTP_ADDRESS","valueFrom":{"configMapKeyRef":
{"key":"address","name":"smtp"}}},{"name":"SMTP_USER_NAME","valueFrom":
{"configMapKeyRef":{"key":"username","name":"smtp"}}},

{"name":"SMTP_PASSWORD","valueFrom":{"configMapKeyRef":
{"key":"password","name":"smtp"}}},{"name":"SMTP_DOMAIN","valueFrom":
{"configMapKeyRef":{"key":"domain","name":"smtp"}}},{"name":"SMTP_PORT","valueFrom":
{"configMapKeyRef":{"key":"port","name":"smtp"}}},
{"name":"SMTP_AUTHENTICATION","valueFrom":{"configMapKeyRef":
{"key":"authentication","name":"smtp"}}},
{"name":"SMTP_OPENSSL_VERIFY_MODE","valueFrom":{"configMapKeyRef":
{"key":"openssl.verify.mode","name":"smtp"}}},
{"name":"APICAST_ACCESS_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ACCESS_TOKEN","name":"system-master-apicast"}}},
{"name":"ZYNC_AUTHENTICATION_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ZYNC_AUTHENTICATION_TOKEN","name":"zync"}}},
{"name":"CONFIG_INTERNAL_API_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"username","name":"backend-internal-api"}}},
{"name":"CONFIG_INTERNAL_API_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"password","name":"backend-internal-api"}}}]}}},"template":{"spec":{"containers":
[{"name":"system-master","env":[{"name":"AMP_RELEASE","value":null,"valueFrom":
{"configMapKeyRef":{"key":"AMP_RELEASE","name":"system-environment"}}},
{"name":"APICAST_REGISTRY_URL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"APICAST_REGISTRY_URL","name":"system-environment"}}},
{"name":"FORCE_SSL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"FORCE_SSL","name":"system-environment"}}},
{"name":"PROVIDER_PLAN","value":null,"valueFrom":{"configMapKeyRef":
{"key":"PROVIDER_PLAN","name":"system-environment"}}},
{"name":"RAILS_ENV","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_ENV","name":"system-environment"}}},
{"name":"RAILS_LOG_LEVEL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_LOG_LEVEL","name":"system-environment"}}},
{"name":"RAILS_LOG_TO_STDOUT","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_LOG_TO_STDOUT","name":"system-environment"}}},
{"name":"SSL_CERT_DIR","value":null,"valueFrom":{"configMapKeyRef":
{"key":"SSL_CERT_DIR","name":"system-environment"}}},
{"name":"THINKING_SPHINX_PORT","value":null,"valueFrom":{"configMapKeyRef":
{"key":"THINKING_SPHINX_PORT","name":"system-environment"}}},
{"name":"THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE","value":null,"value
From":{"configMapKeyRef":
{"key":"THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE","name":"system-
environment"}}},{"name":"THREESCALE_SUPERDOMAIN","value":null,"valueFrom":
{"configMapKeyRef":{"key":"THREESCALE_SUPERDOMAIN","name":"system-
environment"}}},{"name":"DATABASE_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-database"}}},
{"name":"MASTER_DOMAIN","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_DOMAIN","name":"system-seed"}}},
{"name":"MASTER_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_USER","name":"system-seed"}}},
{"name":"MASTER_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_PASSWORD","name":"system-seed"}}},
{"name":"ADMIN_ACCESS_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_ACCESS_TOKEN","name":"system-seed"}}},
{"name":"USER_LOGIN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_USER","name":"system-seed"}}},
{"name":"USER_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_PASSWORD","name":"system-seed"}}},
{"name":"TENANT_NAME","value":null,"valueFrom":{"secretKeyRef":
{"key":"TENANT_NAME","name":"system-seed"}}},
{"name":"EVENTS_SHARED_SECRET","value":null,"valueFrom":{"secretKeyRef":

{"key":"PASSWORD","name":"system-events-hook"}}},
{"name":"RECAPTCHA_PUBLIC_KEY","value":null,"valueFrom":{"secretKeyRef":
{"key":"PUBLIC_KEY","name":"system-recaptcha"}}},
{"name":"RECAPTCHA_PRIVATE_KEY","value":null,"valueFrom":{"secretKeyRef":
{"key":"PRIVATE_KEY","name":"system-recaptcha"}}},
{"name":"SECRET_KEY_BASE","value":null,"valueFrom":{"secretKeyRef":
{"key":"SECRET_KEY_BASE","name":"system-app"}}},
{"name":"REDIS_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-redis"}}},
{"name":"MEMCACHE_SERVERS","value":null,"valueFrom":{"secretKeyRef":
{"key":"SERVERS","name":"system-memcache"}}},
{"name":"BACKEND_REDIS_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_URL","name":"backend-redis"}}},
{"name":"APICAST_BACKEND_ROOT_ENDPOINT","value":null,"valueFrom":
{"secretKeyRef":{"key":"route_endpoint","name":"backend-listener"}}},
{"name":"BACKEND_ROUTE","value":null,"valueFrom":{"secretKeyRef":
{"key":"route_endpoint","name":"backend-listener"}}},
{"name":"APICAST_ACCESS_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ACCESS_TOKEN","name":"system-master-apicast"}}},
{"name":"ZYNC_AUTHENTICATION_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ZYNC_AUTHENTICATION_TOKEN","name":"zync"}}},
{"name":"CONFIG_INTERNAL_API_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"username","name":"backend-internal-api"}}},
{"name":"CONFIG_INTERNAL_API_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"password","name":"backend-internal-api"}}}]},{"name":"system-provider","env":
[{"name":"AMP_RELEASE","value":null,"valueFrom":{"configMapKeyRef":
{"key":"AMP_RELEASE","name":"system-environment"}}},
{"name":"APICAST_REGISTRY_URL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"APICAST_REGISTRY_URL","name":"system-environment"}}},
{"name":"FORCE_SSL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"FORCE_SSL","name":"system-environment"}}},
{"name":"PROVIDER_PLAN","value":null,"valueFrom":{"configMapKeyRef":
{"key":"PROVIDER_PLAN","name":"system-environment"}}},
{"name":"RAILS_ENV","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_ENV","name":"system-environment"}}},
{"name":"RAILS_LOG_LEVEL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_LOG_LEVEL","name":"system-environment"}}},
{"name":"RAILS_LOG_TO_STDOUT","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_LOG_TO_STDOUT","name":"system-environment"}}},
{"name":"SSL_CERT_DIR","value":null,"valueFrom":{"configMapKeyRef":
{"key":"SSL_CERT_DIR","name":"system-environment"}}},
{"name":"THINKING_SPHINX_PORT","value":null,"valueFrom":{"configMapKeyRef":
{"key":"THINKING_SPHINX_PORT","name":"system-environment"}}},
{"name":"THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE","value":null,"value
From":{"configMapKeyRef":
{"key":"THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE","name":"system-
environment"}}},{"name":"THREESCALE_SUPERDOMAIN","value":null,"valueFrom":
{"configMapKeyRef":{"key":"THREESCALE_SUPERDOMAIN","name":"system-
environment"}}},{"name":"DATABASE_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-database"}}},
{"name":"MASTER_DOMAIN","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_DOMAIN","name":"system-seed"}}},
{"name":"MASTER_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_USER","name":"system-seed"}}},
{"name":"MASTER_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_PASSWORD","name":"system-seed"}}},

{"name":"ADMIN_ACCESS_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_ACCESS_TOKEN","name":"system-seed"}}},
{"name":"USER_LOGIN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_USER","name":"system-seed"}}},
{"name":"USER_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_PASSWORD","name":"system-seed"}}},
{"name":"TENANT_NAME","value":null,"valueFrom":{"secretKeyRef":
{"key":"TENANT_NAME","name":"system-seed"}}},
{"name":"EVENTS_SHARED_SECRET","value":null,"valueFrom":{"secretKeyRef":
{"key":"PASSWORD","name":"system-events-hook"}}},
{"name":"RECAPTCHA_PUBLIC_KEY","value":null,"valueFrom":{"secretKeyRef":
{"key":"PUBLIC_KEY","name":"system-recaptcha"}}},
{"name":"RECAPTCHA_PRIVATE_KEY","value":null,"valueFrom":{"secretKeyRef":
{"key":"PRIVATE_KEY","name":"system-recaptcha"}}},
{"name":"SECRET_KEY_BASE","value":null,"valueFrom":{"secretKeyRef":
{"key":"SECRET_KEY_BASE","name":"system-app"}}},
{"name":"REDIS_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-redis"}}},
{"name":"MEMCACHE_SERVERS","value":null,"valueFrom":{"secretKeyRef":
{"key":"SERVERS","name":"system-memcache"}}},
{"name":"BACKEND_REDIS_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_URL","name":"backend-redis"}}},
{"name":"APICAST_BACKEND_ROOT_ENDPOINT","value":null,"valueFrom":
{"secretKeyRef":{"key":"route_endpoint","name":"backend-listener"}}},
{"name":"BACKEND_ROUTE","value":null,"valueFrom":{"secretKeyRef":
{"key":"route_endpoint","name":"backend-listener"}}},
{"name":"APICAST_ACCESS_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ACCESS_TOKEN","name":"system-master-apicast"}}},
{"name":"ZYNC_AUTHENTICATION_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ZYNC_AUTHENTICATION_TOKEN","name":"zync"}}},
{"name":"CONFIG_INTERNAL_API_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"username","name":"backend-internal-api"}}},
{"name":"CONFIG_INTERNAL_API_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"password","name":"backend-internal-api"}}}]},{"name":"system-developer","env":
[{"name":"AMP_RELEASE","value":null,"valueFrom":{"configMapKeyRef":
{"key":"AMP_RELEASE","name":"system-environment"}}},
{"name":"APICAST_REGISTRY_URL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"APICAST_REGISTRY_URL","name":"system-environment"}}},
{"name":"FORCE_SSL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"FORCE_SSL","name":"system-environment"}}},
{"name":"PROVIDER_PLAN","value":null,"valueFrom":{"configMapKeyRef":
{"key":"PROVIDER_PLAN","name":"system-environment"}}},
{"name":"RAILS_ENV","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_ENV","name":"system-environment"}}},
{"name":"RAILS_LOG_LEVEL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_LOG_LEVEL","name":"system-environment"}}},
{"name":"RAILS_LOG_TO_STDOUT","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_LOG_TO_STDOUT","name":"system-environment"}}},
{"name":"SSL_CERT_DIR","value":null,"valueFrom":{"configMapKeyRef":
{"key":"SSL_CERT_DIR","name":"system-environment"}}},
{"name":"THINKING_SPHINX_PORT","value":null,"valueFrom":{"configMapKeyRef":
{"key":"THINKING_SPHINX_PORT","name":"system-environment"}}},
{"name":"THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE","value":null,"value
From":{"configMapKeyRef":
{"key":"THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE","name":"system-
environment"}}},{"name":"THREESCALE_SUPERDOMAIN","value":null,"valueFrom":

```
{"configMapKeyRef":{"key":"THREESCALE_SUPERDOMAIN","name":"system-
environment"}}},{"name":"DATABASE_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-database"}}},
{"name":"MASTER_DOMAIN","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_DOMAIN","name":"system-seed"}}},
{"name":"MASTER_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_USER","name":"system-seed"}}},
{"name":"MASTER_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_PASSWORD","name":"system-seed"}}},
{"name":"ADMIN_ACCESS_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_ACCESS_TOKEN","name":"system-seed"}}},
{"name":"USER_LOGIN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_USER","name":"system-seed"}}},
{"name":"USER_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_PASSWORD","name":"system-seed"}}},
{"name":"TENANT_NAME","value":null,"valueFrom":{"secretKeyRef":
{"key":"TENANT_NAME","name":"system-seed"}}},
{"name":"EVENTS_SHARED_SECRET","value":null,"valueFrom":{"secretKeyRef":
{"key":"PASSWORD","name":"system-events-hook"}}},
{"name":"RECAPTCHA_PUBLIC_KEY","value":null,"valueFrom":{"secretKeyRef":
{"key":"PUBLIC_KEY","name":"system-recaptcha"}}},
{"name":"RECAPTCHA_PRIVATE_KEY","value":null,"valueFrom":{"secretKeyRef":
{"key":"PRIVATE_KEY","name":"system-recaptcha"}}},
{"name":"SECRET_KEY_BASE","value":null,"valueFrom":{"secretKeyRef":
{"key":"SECRET_KEY_BASE","name":"system-app"}}},
{"name":"REDIS_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-redis"}}},
{"name":"MEMCACHE_SERVERS","value":null,"valueFrom":{"secretKeyRef":
{"key":"SERVERS","name":"system-memcache"}}},
{"name":"BACKEND_REDIS_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_URL","name":"backend-redis"}}},
{"name":"APICAST_BACKEND_ROOT_ENDPOINT","value":null,"valueFrom":
{"secretKeyRef":{"key":"route_endpoint","name":"backend-listener"}}},
{"name":"BACKEND_ROUTE","value":null,"valueFrom":{"secretKeyRef":
{"key":"route_endpoint","name":"backend-listener"}}},
{"name":"APICAST_ACCESS_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ACCESS_TOKEN","name":"system-master-apicast"}}},
{"name":"ZYNC_AUTHENTICATION_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ZYNC_AUTHENTICATION_TOKEN","name":"zync"}}},
{"name":"CONFIG_INTERNAL_API_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"username","name":"backend-internal-api"}}},
{"name":"CONFIG_INTERNAL_API_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"password","name":"backend-internal-api"}}}]}],"volumes":[{"configMap":
{"defaultMode":420,"items":[{"key":"zync.yml","path":"zync.yml"},
{"key":"rolling_updates.yml","path":"rolling_updates.yml"},
{"key":"service_discovery.yml","path":"service_discovery.yml"}],"name":"system"},"name":"syste
m-config"}]}}}}'
```

5. Update the **system-sidekiq** DeploymentConfig to gather the database information of *system* from the new secret:

```
oc patch dc/system-sidekiq -p '{"spec":{"template":{"spec":{"containers":[{"name":"system-
sidekiq","env":[{"name":"AMP_RELEASE","value":null,"valueFrom":{"configMapKeyRef":
{"key":"AMP_RELEASE","name":"system-environment"}}},
{"name":"APICAST_REGISTRY_URL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"APICAST_REGISTRY_URL","name":"system-environment"}}},
```

{"name":"FORCE_SSL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"FORCE_SSL","name":"system-environment"}}},
{"name":"PROVIDER_PLAN","value":null,"valueFrom":{"configMapKeyRef":
{"key":"PROVIDER_PLAN","name":"system-environment"}}},
{"name":"RAILS_ENV","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_ENV","name":"system-environment"}}},
{"name":"RAILS_LOG_LEVEL","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_LOG_LEVEL","name":"system-environment"}}},
{"name":"RAILS_LOG_TO_STDOUT","value":null,"valueFrom":{"configMapKeyRef":
{"key":"RAILS_LOG_TO_STDOUT","name":"system-environment"}}},
{"name":"SSL_CERT_DIR","value":null,"valueFrom":{"configMapKeyRef":
{"key":"SSL_CERT_DIR","name":"system-environment"}}},
{"name":"THINKING_SPHINX_PORT","value":null,"valueFrom":{"configMapKeyRef":
{"key":"THINKING_SPHINX_PORT","name":"system-environment"}}},
{"name":"THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE","value":null,"value
From":{"configMapKeyRef":
{"key":"THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE","name":"system-
environment"}}},{"name":"THREESCALE_SUPERDOMAIN","value":null,"valueFrom":
{"configMapKeyRef":{"key":"THREESCALE_SUPERDOMAIN","name":"system-
environment"}}},{"name":"DATABASE_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-database"}}},
{"name":"MASTER_DOMAIN","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_DOMAIN","name":"system-seed"}}},
{"name":"MASTER_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_USER","name":"system-seed"}}},
{"name":"MASTER_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"MASTER_PASSWORD","name":"system-seed"}}},
{"name":"ADMIN_ACCESS_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_ACCESS_TOKEN","name":"system-seed"}}},
{"name":"USER_LOGIN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_USER","name":"system-seed"}}},
{"name":"USER_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"ADMIN_PASSWORD","name":"system-seed"}}},
{"name":"TENANT_NAME","value":null,"valueFrom":{"secretKeyRef":
{"key":"TENANT_NAME","name":"system-seed"}}},
{"name":"EVENTS_SHARED_SECRET","value":null,"valueFrom":{"secretKeyRef":
{"key":"PASSWORD","name":"system-events-hook"}}},
{"name":"RECAPTCHA_PUBLIC_KEY","value":null,"valueFrom":{"secretKeyRef":
{"key":"PUBLIC_KEY","name":"system-recaptcha"}}},
{"name":"RECAPTCHA_PRIVATE_KEY","value":null,"valueFrom":{"secretKeyRef":
{"key":"PRIVATE_KEY","name":"system-recaptcha"}}},
{"name":"SECRET_KEY_BASE","value":null,"valueFrom":{"secretKeyRef":
{"key":"SECRET_KEY_BASE","name":"system-app"}}},
{"name":"REDIS_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-redis"}}},
{"name":"MEMCACHE_SERVERS","value":null,"valueFrom":{"secretKeyRef":
{"key":"SERVERS","name":"system-memcache"}}},
{"name":"BACKEND_REDIS_URL","value":null,"valueFrom":{"secretKeyRef":
{"key":"REDIS_STORAGE_URL","name":"backend-redis"}}},
{"name":"APICAST_BACKEND_ROOT_ENDPOINT","value":null,"valueFrom":
{"secretKeyRef":{"key":"route_endpoint","name":"backend-listener"}}},
{"name":"BACKEND_ROUTE","value":null,"valueFrom":{"secretKeyRef":
{"key":"route_endpoint","name":"backend-listener"}}},
{"name":"APICAST_ACCESS_TOKEN","value":null,"valueFrom":{"secretKeyRef":
{"key":"ACCESS_TOKEN","name":"system-master-apicast"}}},
{"name":"ZYNC_AUTHENTICATION_TOKEN","value":null,"valueFrom":{"secretKeyRef":

```
{"key":"ZYNC_AUTHENTICATION_TOKEN","name":"zync"}}},
{"name":"CONFIG_INTERNAL_API_USER","value":null,"valueFrom":{"secretKeyRef":
{"key":"username","name":"backend-internal-api"}}},
{"name":"CONFIG_INTERNAL_API_PASSWORD","value":null,"valueFrom":{"secretKeyRef":
{"key":"password","name":"backend-internal-api"}}}]}],"initContainers":[{"name":"check-
svc","command":["bash","-c","bundle exec sh -c \"until rake boot:redis && curl --output
/dev/null --silent --fail --head http://system-master:3000/status; do sleep
$SLEEP_SECONDS; done\""],"env":[{"name":"REDIS_URL","value":null,"valueFrom":
{"secretKeyRef":{"key":"URL","name":"system-redis"}}}]}],"volumes":[{"configMap":
{"defaultMode":420,"items":[{"key":"zync.yml","path":"zync.yml"},
{"key":"rolling_updates.yml","path":"rolling_updates.yml"},
{"key":"service_discovery.yml","path":"service_discovery.yml"}]},"name":"system-config"}]}}}}'
```

6. Update the **system-sphinx** DeploymentConfig to gather the database information of *system* from the new secret:

```
oc patch dc/system-sphinx -p '{"spec":{"template":{"spec":{"containers":[{"name":"system-
sphinx","env":[{"name":"DATABASE_URL","value":"","valueFrom":{"secretKeyRef":
{"key":"URL","name":"system-database"}}}]}]}}}}'
```

7. Patch the **apicast-staging** DeploymentConfig containers:

```
oc patch dc/apicast-staging -p '{"spec":{"template": {"spec": { "containers": [ {"name":
"apicast-staging", "env": [ { "name": "THREESCALE_PORTAL_ENDPOINT",
"value":null,"valueFrom": { "secretKeyRef": { "key": "PROXY_CONFIGS_ENDPOINT",
"name": "system-master-apicast" } } }, { "name": "BACKEND_ENDPOINT_OVERRIDE",
"value":null,"valueFrom": { "secretKeyRef": { "key": "service_endpoint", "name": "backend-
listener" } } }, { "name": "APICAST_MANAGEMENT_API", "value":null,"valueFrom": {
"configMapKeyRef": { "key": "APICAST_MANAGEMENT_API", "name": "apicast-
environment" } } }, { "name": "OPENSSL_VERIFY", "value":null,"valueFrom": {
"configMapKeyRef": { "key": "OPENSSL_VERIFY", "name": "apicast-environment" } } }, {
"name": "APICAST_RESPONSE_CODES", "value":null,"valueFrom": { "configMapKeyRef": {
"key": "APICAST_RESPONSE_CODES", "name": "apicast-environment" } } }, { "name":
"APICAST_CONFIGURATION_LOADER", "value": "lazy" }, { "name":
"APICAST_CONFIGURATION_CACHE", "value": "0" }, { "name":
"THREESCALE_DEPLOYMENT_ENV", "value": "staging" }, { "name": "REDIS_URL",
"value":null,"valueFrom": { "secretKeyRef": { "key": "STAGING_URL", "name": "apicast-redis"
} } } ]}]}}}}'
```

8. Patch the **apicast-production** DeploymentConfig containers:

```
oc patch dc/apicast-production -p '{"spec":{"template":{"spec": { "containers": [
{"name":"apicast-production", "env": [ { "name": "THREESCALE_PORTAL_ENDPOINT",
"value":null,"valueFrom": { "secretKeyRef": { "key": "PROXY_CONFIGS_ENDPOINT",
"name": "system-master-apicast" } } }, { "name": "BACKEND_ENDPOINT_OVERRIDE",
"value":null,"valueFrom": { "secretKeyRef": { "key": "service_endpoint", "name": "backend-
listener" } } }, { "name": "APICAST_MANAGEMENT_API", "value":null,"valueFrom": {
"configMapKeyRef": { "key": "APICAST_MANAGEMENT_API", "name": "apicast-
environment" } } }, { "name": "OPENSSL_VERIFY", "value":null,"valueFrom": {
"configMapKeyRef": { "key": "OPENSSL_VERIFY", "name": "apicast-environment" } } }, {
"name": "APICAST_RESPONSE_CODES", "value":null,"valueFrom": { "configMapKeyRef": {
"key": "APICAST_RESPONSE_CODES", "name": "apicast-environment" } } }, { "name":
"APICAST_CONFIGURATION_LOADER", "value": "boot" }, { "name":
"APICAST_CONFIGURATION_CACHE", "value": "300" }, { "name":
```

```
"THREESCALE_DEPLOYMENT_ENV", "value": "production" }, { "name": "REDIS_URL",
"value":null,"valueFrom": { "secretKeyRef": { "key": "PRODUCTION_URL", "name": "apicast-
redis" } } } ]}]}}}}'
```

9. Patch the **apicast-wildcard-router** DeploymentConfig containers:

```
oc patch dc/apicast-wildcard-router -p '{"spec":{"template":{"spec":{"containers":
[{"name":"apicast-wildcard-router","env":[{"name":"API_HOST","value":null,"valueFrom":
{"secretKeyRef":{"key":"BASE_URL","name":"system-master-apicast"}}}]}]}}}}'
```

1.2.6. Patching image streams

1. Patch the **amp-system** image stream:

   - If 3scale API Management is deployed with **Oracle Database**,

     a. Update the system image 2.4.0 image stream:

     ```
     oc patch imagestream/amp-system --type=json -p '[{"op": "add", "path": "/spec/tags/-
     ", "value": {"annotations": {"openshift.io/display-name": "AMP system 2.4.0"}, "from": {
     "kind": "DockerImage", "name": "registry.access.redhat.com/3scale-amp24/system"},
     "name": "2.4.0", "referencePolicy": {"type": "Source"}}}]'
     ```

     b. Update the build configuration for **3scale-amp-oracle** to fetch from the **2.4** tag:

     ```
     oc patch bc 3scale-amp-system-oracle --type json -p '[{"op": "replace", "path":
     "/spec/strategy/dockerStrategy/from/name", "value": "amp-system:2.4.0"}]'
     ```

     c. Run the build:

     ```
     oc start-build 3scale-amp-system-oracle --from-dir=.
     ```

   - If 3scale API Management is deployed with a different database, use the following commands:

     ```
     oc patch imagestream/amp-system --type=json -p '[{"op": "add", "path": "/spec/tags/-",
     "value": {"annotations": {"openshift.io/display-name": "AMP system 2.4.0"}, "from": {
     "kind": "DockerImage", "name": "registry.access.redhat.com/3scale-amp24/system"},
     "name": "2.4.0", "referencePolicy": {"type": "Source"}}}]'
     oc patch imagestream/amp-system --type=json -p '[{"op": "add", "path": "/spec/tags/-",
     "value": {"annotations": {"openshift.io/display-name": "AMP system (latest)"}, "from": {
     "kind": "ImageStreamTag", "name": "2.4.0"}, "name": "latest", "referencePolicy": {"type":
     "Source"}}}]'
     ```

2. Patch the **amp-apicast** image stream:

```
oc patch imagestream/amp-apicast --type=json -p '[{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "AMP APIcast 2.4.0"}, "from": { "kind":
"DockerImage", "name": "registry.access.redhat.com/3scale-amp24/apicast-gateway"},
"name": "2.4.0", "referencePolicy": {"type": "Source"}}}]'
oc patch imagestream/amp-apicast --type=json -p '[{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "AMP APIcast (latest)"}, "from": { "kind":
"ImageStreamTag", "name": "2.4.0"}, "name": "latest", "referencePolicy": {"type": "Source"}}}]'
```

3. Patch the **amp-backend** image stream:

```
oc patch imagestream/amp-backend --type=json -p '[{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "AMP Backend 2.4.0"}, "from": { "kind":
"DockerImage", "name": "registry.access.redhat.com/3scale-amp24/backend"}, "name":
"2.4.0", "referencePolicy": {"type": "Source"}}}]'
oc patch imagestream/amp-backend --type=json -p '[{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "AMP Backend (latest)"}, "from": {
"kind": "ImageStreamTag", "name": "2.4.0"}, "name": "latest", "referencePolicy": {"type":
"Source"}}}]'
```

4. Patch the **amp-zync** image stream:

```
oc patch imagestream/amp-zync --type=json -p '[{"op": "add", "path": "/spec/tags/-", "value":
{"annotations": {"openshift.io/display-name": "AMP Zync 2.4.0"}, "from": { "kind":
"DockerImage", "name": "registry.access.redhat.com/3scale-amp24/zync"}, "name": "2.4.0",
"referencePolicy": {"type": "Source"}}}]'
oc patch imagestream/amp-zync --type=json -p '[{"op": "add", "path": "/spec/tags/-", "value":
{"annotations": {"openshift.io/display-name": "AMP Zync (latest)"}, "from": { "kind":
"ImageStreamTag", "name": "2.4.0"}, "name": "latest", "referencePolicy": {"type": "Source"}}}]'
```

1.2.7. Adding the entry for Service Discovery

As part of the upgrade process, add the ConfigMap entry for Service Discovery. Service Discovery is a feature that allows you to add APIs for management by recognizing the discoverable running services in an OpenShift cluster

1. Edit the **system** ConfigMap to add the entry:

```
oc edit configmap system
```

2. Add the entry:

```
service_discovery.yml: |
    production:
      enabled: <%= cluster_token_file_exists = File.exists?(cluster_token_file_path =
'/var/run/secrets/kubernetes.io/serviceaccount/token') %>
      server_scheme: 'https'
      server_host: 'kubernetes.default.svc.cluster.local'
      server_port: 443
      bearer_token: "<%= File.read(cluster_token_file_path) if cluster_token_file_exists %>"
      authentication_method: service_account # can be service_account|oauth
      oauth_server_type: builtin # can be builtin|rh_sso
      client_id:
      client_secret:
      timeout: 1
      open_timeout: 1
      max_retry: 5
      verify_ssl: <%= OpenSSL::SSL::VERIFY_NONE %> # 0
```

You can continue with additional configurations to import services by referring to the Service Discovery guide.

1.2.8. Configuring additional DeploymentConfigs

1. Patch the **system-memcache** DeploymentConfig:

```
oc patch dc/system-memcache --patch='{"spec":{"template":{"spec":{"containers":[{"name":
"memcache", "image":"registry.access.redhat.com/3scale-amp20/memcached"}]}}}}'
```

2. Delete the **system-resque** DeploymentConfig:

```
oc delete dc/system-resque
```

3. Set environment variables to increase **system-sidekiq** concurrency to the recommended levels:

```
oc set env dc/system-sidekiq RAILS_MAX_THREADS=25
```

4. Set environment variable to update the visible release version:

```
oc set env dc/system-app AMP_RELEASE=2.4.0
```

1.3. CHANGING ADMINISTRATOR IMPERSONATION (OPTIONAL)

As 3scale API Management is open source, impersonation data is publicly disclosed. For this reason, you might want to change some data:

- The unique username for the impersonation of administrators.
- The domain of the email of the impersonation for the administrator user.

As an example, assume that **username:<your-username>** and **domain:<example.com>**. To change the impersonation of the administrator, you need to follow these steps:

1. Create a file locally called **system-impersonation-secret.yml** with the following content:

```
cat > system-impersonation-secret.yml <<EOF

apiVersion: v1
kind: Secret
metadata:
  creationTimestamp: null
  labels:
    3scale.component: system
    app: ${APP_LABEL}
  name: system-impersonation
stringData:
  username: "<your-username>"
  domain: "<example.com>"
type: Opaque
EOF
```

2. Change **<your-username>** and **<example.com>** to the chosen user name and domain.

3. Create a secret:

```
oc create -f system-impersonation-secret.yml
```

4. Set the environment variables from this secret with:

```
oc set env --from=secret/system-impersonation --prefix=IMPERSONATION_ADMIN
dc/system-app
```

5. Redeploy **system-app**:

```
oc rollout latest system-app
```

6. Connect to the **system-master** container of **system-app** deployment:

```
oc rsh -c system-master dc/system-app
```

7. In this container execute, changing **<your-username>** and **<example.com>** accordingly:

```
bundle exec rake "impersonation_admin_user:update[<your-username>,<example.com>]"
```

You should be able to impersonate a tenant from the user interface now.

CHAPTER 2. BUILDING A 3SCALE API MANAGEMENT SYSTEM IMAGE WITH THE ORACLE DATABASE RELATIONAL DATABASE MANAGEMENT SYSTEM

By default, 3scale has a component called **system** which stores configuration data in a MySQL database. You have the option to override the default database and store your information in an external Oracle Database. Follow the steps in this document to build a custom system container image with your own Oracle Database client binaries and deploy 3scale to OpenShift.

2.1. BEFORE YOU BEGIN

2.1.1. Obtain Oracle software components

Before you can build the custom 3scale system container image, you must acquire a supported version of the following Oracle software components:

- Oracle Instant Client Package Basic or Basic Light
- Oracle Instant Client Package SDK
- Oracle Instant Client Package ODBC

2.1.2. Meet prerequisites

You must also meet the following prerequisites:

- A supported version of Oracle Database accessible from your OpenShift cluster
- Access to the Oracle Database **system** user for installation procedures
- Possess the Red Hat 3scale 2.4 amp.yml template

2.2. PREPARING ORACLE DATABASE

1. Create a new database
   The following settings are required for the Oracle Database to work with 3scale:

   ```
   ALTER SYSTEM SET max_string_size=extended SCOPE=SPFILE;
   ```

   ```
   ALTER SYSTEM SET compatible='12.2.0.1' SCOPE=SPFILE;
   ```

2. Collect the database details.
   Get the following information that will be needed for 3scale configuration:

   - Oracle Database URL
   - Oracle Database service name
   - Oracle Database **system** user name and password
   - Oracle Database service name

For information on creating a new database in Oracle Database, refer to the Oracle documentation.

2.3. BUILDING THE SYSTEM IMAGE

1. clone the 3scale-amp-openshift-templates github repository
2. place your Oracle Database Instant Client Package files into the **3scale-amp-openshift-templates/amp/system-oracle/oracle-client-files** directory
3. run the **oc new-app** command with the -f option and specify the **build.yml** OpenShift template

```
$ oc new-app -f build.yml
```

4. run the **oc new-app** command with the -f option, specifying the **amp.yml** OpenShift template, and the -p option, specifying the **WILDCARD_DOMAIN** parameter with the domain of your OpenShift cluster

```
$ oc new-app -f amp.yml -p WILDCARD_DOMAIN=example.com
```

5. enter the following shell **for** loop command, specifying the following information you collected in the Preparing Oracle Database section previously:
   - **{USER}**: the username that will represent 3scale in your Oracle Database
   - **{PASSWORD}**: the password for **USER**
   - **{ORACLE_DB_URL}**: the URL of your Oracle Database
   - **{DATABASE}**: the service name of the database you created in Oracle Database
   - **{PORT}**: the port number of your Oracle Database

```
for dc in system-app system-resque system-sidekiq system-sphinx; do oc env dc/$dc --overwrite DATABASE_URL="oracle-enhanced://{USER}:{PASSWORD}@{ORACLE_DB_URL}:{PORT}/{DATABASE}"; done
```

6. enter the following **oc patch** command, specifying the same **USER**, **PASSWORD**, **ORACLE_DB_URL**, **PORT**, and **DATABASE** values that you provided in the previous step above:

```
$ oc patch dc/system-app -p '[{"op": "replace", "path": "/spec/strategy/rollingParams/pre/execNewPod/env/1/value", "value": "oracle-enhanced://{USER}:{PASSWORD}@{ORACLE_DB_URL}:{PORT}/{DATABASE}"}]' --type=json
```

7. enter the following **oc patch** command, specifying your own Oracle Database **system** user password in the **SYSTEM_PASSWORD** field:

```
$ oc patch dc/system-app -p '[{"op": "add", "path": "/spec/strategy/rollingParams/pre/execNewPod/env/-", "value": {"name": "ORACLE_SYSTEM_PASSWORD", "value": "SYSTEM_PASSWORD"}}]' --type=json
```

8. enter the **oc start-build** command to build the new system image:

```
oc start-build 3scale-amp-system-oracle --from-dir=.
```

CHAPTER 3. 3SCALE API MANAGEMENT ON-PREMISES INSTALLATION GUIDE

This guide walks you through steps to install Red Hat 3scale API Management 2.4 (on-premises) on OpenShift using OpenShift templates.

3.1. PREREQUISITES

- You must configure 3scale API Management servers for UTC (Coordinated Universal Time).
- You must configure a valid HTTPS certificate (not self-signed) on OpenShift. For more information, see Create Secure Routes on OpenShift .

3.2. 3SCALE OPENSHIFT TEMPLATES

3scale API Management 2.4 provides an OpenShift template. You can use this template to deploy 3scale API Management onto OpenShift Container Platform.

The 3scale API Management template is composed of the following:

- Two built-in APIcast API gateways
- One 3scale API Management Admin Portal and developer portal with persistent storage

3.3. SYSTEM REQUIREMENTS

This section lists the requirements for the 3scale API Management OpenShift template.

3.3.1. Environment requirements

3scale API Management requires an environment specified in supported configurations.

Persistent Volumes:

- 3 RWO (ReadWriteOnce) persistent volumes for Redis and MySQL persistence
- 1 RWX (ReadWriteMany) persistent volume for CMS and System-app Assets

The RWX persistent volume must be configured to be group writable. For a list of persistent volume types that support the required access modes, see the OpenShift documentation .

3.3.2. Hardware requirements

Hardware requirements depend on your usage needs. Red Hat recommends that you test and configure your environment to meet your specific requirements. Following are the recommendations when configuring your environment for 3scale on OpenShift:

- Compute optimized nodes for deployments on cloud environments (AWS c4.2xlarge or Azure Standard_F8).
- Very large installations may require a separate node (AWS M4 series or Azure Av2 series) for Redis if memory requirements exceed your current node's available RAM.
- Separate nodes between routing and compute tasks.

- Dedicated compute nodes to 3scale specific tasks.
- Set the **PUMA_WORKERS** variable of the backend listener to the number of cores in your compute node.

3.4. CONFIGURE NODES AND ENTITLEMENTS

Before you can deploy 3scale on OpenShift, you must configure your nodes and the entitlements required for your environment to fetch images from Red Hat.

Perform the following steps to configure the entitlements:

1. Install Red Hat Enterprise Linux (RHEL) on each of your nodes.
2. Register your nodes with Red Hat using the Red Hat Subscription Manager (RHSM), via the interface or the command line.
3. Attach your nodes to your 3scale subscription using RHSM.
4. Install OpenShift on your nodes, complying with the following requirements:
   - Use a supported OpenShift version.
   - Configure persistent storage on a file system that supports multiple writes.
5. Install the OpenShift command line interface.
6. Enable access to the **rhel-7-server-3scale-amp-2.4-rpms** repository using the subscription manager:

```
sudo subscription-manager repos --enable=rhel-7-server-3scale-amp-2.4-rpms
```

7. Install the **3scale-amp-template**. The template will be saved at **/opt/amp/templates**.

```
sudo yum install 3scale-amp-template
```

3.5. DEPLOY 3SCALE ON OPENSHIFT USING A TEMPLATE

3.5.1. Prerequisites

- An OpenShift cluster configured as specified in the Chapter 3, *Configure Nodes and Entitlements* section.
- A domain, preferably wildcard, that resolves to your OpenShift cluster.
- Access to the Red Hat container catalog.
- (Optional) A working SMTP server for email functionality.

Follow these procedures to install 3scale API Management on OpenShift using a **.yml** template:

- Import the 3scale API Management Template
- Configure SMTP Variables (Optional)

3.5.2. Import the 3scale template

Perform the following steps to import the 3scale API Management template into your OpenShift cluster:

1. From a terminal session log in to OpenShift as the cluster administrator:

```
oc login
```

2. Select your project, or create a new project:

```
oc project <project_name>
```

```
oc new-project <project_name>
```

3. Enter the **oc new-app** command:

   a. Specify the **--file** option with the path to the amp.yml file you downloaded as part of the configure nodes and entitlements section .

   b. Specify the **--param** option with the **WILDCARD_DOMAIN** parameter set to the domain of your OpenShift cluster.

   c. Optionally, specify the **--param** option with the **WILDCARD_POLICY** parameter set to **subdomain** to enable wildcard domain routing:
   Without Wildcard Routing:

```
oc new-app --file /opt/amp/templates/amp.yml --param WILDCARD_DOMAIN=
<WILDCARD_DOMAIN>
```

   With Wildcard Routing:

```
oc new-app --file /opt/amp/templates/amp.yml --param WILDCARD_DOMAIN=
<WILDCARD_DOMAIN> --param WILDCARD_POLICY=Subdomain
```

   The terminal shows the master and tenant URLs and credentials for your newly created 3scale API Management Admin Portal. This output should include the following information:

   - master admin username
   - master password
   - master token information
   - tenant username
   - tenant password
   - tenant token information

4. Log in to https://user-admin.3scale-project.example.com as admin/xXxXyz123.

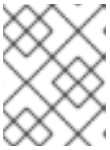
```
* With parameters:
 * ADMIN_PASSWORD=xXxXyz123 # generated
```

```
* ADMIN_USERNAME=admin
* TENANT_NAME=user

* MASTER_NAME=master
* MASTER_USER=master
* MASTER_PASSWORD=xXxXyz123 # generated

--> Success
Access your application via route 'user-admin.3scale-project.example.com'
Access your application via route 'master-admin.3scale-project.example.com'
Access your application via route 'backend-user.3scale-project.example.com'
Access your application via route 'user.3scale-project.example.com'
Access your application via route 'api-user-apicast-staging.3scale-project.example.com'
Access your application via route 'api-user-apicast-production.3scale-project.example.com'
Access your application via route 'apicast-wildcard.3scale-project.example.com'
```

5. Make a note of these details for future reference.



NOTE

You may need to wait a few minutes for 3scale API Management to fully deploy on OpenShift for your login and credentials to work.

More Information

For information about wildcard domains on OpenShift, visit Using Wildcard Routes (for a Subdomain) .

3.5.3. Configure SMTP variables (optional)

OpenShift uses email to send notifications and invite new users. If you intend to use these features, you must provide your own SMTP server and configure SMTP variables in the SMTP config map.

Perform the following steps to configure the SMTP variables in the SMTP config map:

1. If you are not already logged in, log in to OpenShift:

```
oc login
```

   a. Configure variables for the SMTP config map. Use the **oc patch** command, specify the **configmap** and **smtp** objects, followed by the **-p** option and write the following new values in JSON for the following variables:

Variable	Description
address	Allows you to specify a remote mail server as a relay
username	Specify your mail server username
password	Specify your mail server password
domain	Specify a HELO domain

port	Specify the port on which the mail server is listening for new connections
authentication	Specify the authentication type of your mail server. Allowed values: **plain** (sends the password in the clear), **login** (send password Base64 encoded), or **cram_md5** (exchange information and a cryptographic Message Digest 5 algorithm to hash important information)
openssl.verify.mode	Specify how OpenSSL checks certificates when using TLS. Allowed values: **none**, **peer**, **client_once**, or **fail_if_no_peer_cert**.

Example

```
oc patch configmap smtp -p '{"data":{"address":"<your_address>"}}'
oc patch configmap smtp -p '{"data":{"username":"<your_username>"}}'
oc patch configmap smtp -p '{"data":{"password":"<your_password>"}}'
```

2. After you have set the configmap variables, redeploy the **system-app** and **system-sidekiq** pods:

```
oc rollout latest dc/system-app
oc rollout latest dc/system-sidekiq
```

3.6. 3SCALE TEMPLATE PARAMETERS

Template parameters configure environment variables of the 3scale API Management yml template during and after deployment.

Name	Description	Default Value	Required?
APP_LABEL	Used for object app labels	"3scale-api-management"	yes
ZYNC_DATABASE_PASSWORD	Password for the PostgreSQL connection user. Generated randomly if not provided.	N/A	yes
ZYNC_SECRET_KEY_BASE	Secret key base for Zync. Generated randomly if not provided.	N/A	yes

ZYNC_AUTHENTICATION_TOKEN	Authentication token for Zync. Generated randomly if not provided.	N/A	yes
AMP_RELEASE	3scale API Management release tag.	2.4.0	yes
ADMIN_PASSWORD	A randomly generated 3scale API Management administrator account password.	N/A	yes
ADMIN_USERNAME	3scale API Management administrator account username.	admin	yes
APICAST_ACCESS_TOKEN	Read Only Access Token that APIcast will use to download its configuration.	N/A	yes
ADMIN_ACCESS_TOKEN	Admin Access Token with all scopes and write permissions for API access.	N/A	no
WILDCARD_DOMAIN	Root domain for the wildcard routes. For example, a root domain **example.com** will generate **3scale-admin.example.com**.	N/A	yes
WILDCARD_POLICY	Enable wildcard routes to built-in APIcast gateways by setting the value as "Subdomain"	**None**	yes
TENANT_NAME	Tenant name under the root that Admin UI will be available with -admin suffix.	3scale	yes
MYSQL_USER	Username for MySQL user that will be used for accessing the database.	mysql	yes
MYSQL_PASSWORD	Password for the MySQL user.	N/A	yes

MYSQL_DATABASE	Name of the MySQL database accessed.	system	yes
MYSQL_ROOT_PASSWORD	Password for Root user.	N/A	yes
SYSTEM_BACKEND_USERNAME	Internal 3scale API username for internal 3scale api auth.	3scale_api_user	yes
SYSTEM_BACKEND_PASSWORD	Internal 3scale API password for internal 3scale api auth.	N/A	yes
REDIS_IMAGE	Redis image to use	registry.access.redhat.com/rhscl/redis-32-rhel7:3.2	yes
MYSQL_IMAGE	Mysql image to use	registry.access.redhat.com/rhscl/mysql-57-rhel7:5.7	yes
MEMCACHED_IMAGE	Memcached image to use	registry.access.redhat.com/3scale-amp20/memcached:1.4.15	yes
POSTGRESQL_IMAGE	Postgresql image to use	registry.access.redhat.com/rhscl/postgresql-95-rhel7:9.5	yes
AMP_SYSTEM_IMAGE	3scale System image to use	registry.access.redhat.com/3scale-amp24/system	yes
AMP_BACKEND_IMAGE	3scale Backend image to use	registry.access.redhat.com/3scale-amp24/backend	yes
AMP_APICAST_IMAGE	3scale APIcast image to use	registry.access.redhat.com/3scale-amp24/apicast-gateway	yes
AMP_ROUTER_IMAGE	3scale Wildcard Router image to use	registry.access.redhat.com/3scale-amp22/wildcard-router	yes
AMP_ZYNC_IMAGE	3scale Zync image to use	registry.access.redhat.com/3scale-amp24/zync	yes

SYSTEM_BACKEND_SHARED_SECRET	Shared secret to import events from backend to system.	N/A	yes
SYSTEM_APP_SECRET_KEY_BASE	System application secret key base	N/A	yes
APICAST_MANAGEMENT_API	Scope of the APIcast Management API. Can be disabled, status or debug. At least status required for health checks.	status	no
APICAST_OPENSSL_VERIFY	Turn on/off the OpenSSL peer verification when downloading the configuration. Can be set to true/false.	false	no
APICAST_RESPONSE_CODES	Enable logging response codes in APIcast.	true	no
APICAST_REGISTRY_URL	A URL which resolves to the location of APIcast policies	http://apicast-staging:8090/policies	yes
MASTER_USER	Master administrator account username	master	yes
MASTER_NAME	The subdomain value for the master Admin Portal, will be appended with the **-master** suffix	master	yes
MASTER_PASSWORD	A randomly generated master administrator password	N/A	yes
MASTER_ACCESS_TOKEN	A token with master level permissions for API calls	N/A	yes
IMAGESTREAM_TAG_IMPORT_INSECURE	Set to true if the server may bypass certificate verification or connect directly over HTTP during image import.	**false**	yes

3.7. USE APICAST WITH 3SCALE ON OPENSHIFT

APIcast is available with API Manager for 3scale SaaS, and in on-premises installations in OpenShift Container Platform. The configuration procedures are different for both. This section explains how to deploy APIcast with API Manager on OpenShift.

3.7.1. Deploy APIcast templates on an existing OpenShift cluster containing 3scale

3scale API Management OpenShift templates contain two built-in APIcast API gateways by default. If you require more API gateways, or require separate APIcast deployments, you can deploy additional APIcast templates on your OpenShift cluster.

Perform the following steps to deploy additional API gateways on your OpenShift cluster:

1. Create an access token with the following configurations:
   - Scoped to Account Management API
   - Having read-only access
2. Log in to your APIcast Cluster:

```
oc login
```

3. Create a secret that allows APIcast to communicate with 3scale API Management. Specify **new-basicauth**, **apicast-configuration-url-secret**, and the **--password** parameter with the access token, tenant name, and wildcard domain of your 3scale API Management deployment:

```
oc secret new-basicauth apicast-configuration-url-secret --
password=https://<APICAST_ACCESS_TOKEN>@<TENANT_NAME>-admin.
<WILDCARD_DOMAIN>
```

NOTE

TENANT_NAME is the name under the root that the Admin UI will be available with. The default value for **TENANT_NAME** 3scale. If you used a custom value in your 3scale API Management deployment then you must use that value here.

4. Install the APIcast template, **apicast.yml**, on your local machine:

```
sudo yum install 3scale-amp-apicast-gateway-template
```

   This command installs the APIcast template into the directory, /**opt/amp/templates**.

5. Import the APIcast template using running the **oc new-app** command, specifying the **--file** option with the **apicast.yml** file:

```
oc new-app --file /opt/amp/templates/apicast.yml
```

3.7.2. Connect APIcast from an OpenShift cluster outside an OpenShift cluster containing 3scale

If you deploy APIcast on a different OpenShift cluster, outside your 3scale API Management cluster, you must connect over the public route.

1. Create an access token with the following configurations:
   - Scoped to Account Management API
   - Having read-only access
2. Log in to your APIcast Cluster:

```
oc login
```

3. Create a secret that allows APIcast to communicate with 3scale API Management. Specify **new-basicauth**, **apicast-configuration-url-secret**, and the **--password** parameter with the access token, tenant name, and wildcard domain of your 3scale API Management deployment:

```
oc secret new-basicauth apicast-configuration-url-secret --
password=https://<APICAST_ACCESS_TOKEN>@<TENANT_NAME>-admin.
<WILDCARD_DOMAIN>
```

NOTE

TENANT_NAME is the name under the root that the Admin UI will be available with. The default value for`TENANT_NAME` is **3scale**. If you used a custom value in your 3scale API Management deployment then you must use that value here.

4. Deploy APIcast on an OpenShift cluster outside of the OpenShift Cluster with the oc new-app command. Specify the **--file** option and the file path of your **apicast.yml** file:

```
oc new-app --file /path/to/file/apicast.yml
```

5. Update the apicast **BACKEND_ENDPOINT_OVERRIDE** environment variable set to the URL **backend.** followed by the wildcard domain of the OpenShift Cluster containing your 3scale API Management deployment:

```
oc env dc/apicast --overwrite BACKEND_ENDPOINT_OVERRIDE=https://backend-
<TENANT_NAME>.<WILDCARD_DOMAIN>
```

3.7.3. Connect APIcast from other deployments

After you have deployed APIcast on other platforms, you can connect them to 3scale API Management on OpenShift by configuring the **BACKEND_ENDPOINT_OVERRIDE** environment variable in your 3scale API Management OpenShift Cluster:

1. Log in to your 3scale API Management OpenShift Cluster:

```
oc login
```

2. Configure the system-app object **BACKEND_ENDPOINT_OVERRIDE** environment variable:

- If you are using a native installation: BACKEND_ENDPOINT_OVERRIDE=https://backend.<your_openshift_subdomain> bin/apicast
- If are using the Docker containerized environment: docker run -e BACKEND_ENDPOINT_OVERRIDE=https://backend.<your_openshift_subdomain>

3.7.4. Change Built-in APIcast default behavior

In external APIcast deployments, you can modify default behavior by changing the template parameters in the APIcast OpenShift template.

In built-in APIcast deployments, 3scale API Management and APIcast are deployed from a single template. You must modify environment variables after deployment if you wish to change the default behavior for the built-in APIcast deployments.

3.7.5. Connect multiple APIcast deployments on a single OpenShift cluster over internal service routes

If you deploy multiple APIcast gateways into the same OpenShift cluster, you can configure them to connect using internal routes through the backend listener service instead of the default external route configuration.

You must have an OpenShift SDN plugin installed to connect over internal service routes. How you connect depends on which SDN you have installed.

ovs-subnet

If you are using the **ovs-subnet** OpenShift SDN plugin, take the following steps to connect over the internal routes:

1. If not already logged in, log in to your OpenShift Cluster:

```
oc login
```

2. Enter the **oc new-app** command with the path to the **apicast.yml** file:
   a. Specify the **--param** option with the **BACKEND_ENDPOINT_OVERRIDE** parameter set to the domain of your OpenShift cluster's 3scale API Management project:

```
oc new-app -f apicast.yml --param BACKEND_ENDPOINT_OVERRIDE=http://backend-listener.<THREESCALE_PROJECT>.svc.cluster.local:3000
```

ovs-multitenant

If you are using the 'ovs-multitenant' Openshift SDN plugin, take the following steps to connect over the internal routes:

1. If not already logged in, log in to your OpenShift Cluster:

```
oc login
```

2. As admin, specify the **oadm** command with the **pod-network** and **join-projects** options to set up communication between both projects:

```
oadm pod-network join-projects --to=<THREESCALE_PROJECT> <APICAST_PROJECT>
```

3. Enter the **oc new-app** option with the path to the **apicast.yml** file:

    a. Specify the **--param** option with the **BACKEND_ENDPOINT_OVERRIDE** parameter set to the domain of your OpenShift cluster's 3scale API Management project:

```
oc new-app -f apicast.yml --param BACKEND_ENDPOINT_OVERRIDE=http://backend-listener.
<THREESCALE_PROJECT>.svc.cluster.local:3000
```

More information

For information on Openshift SDN and project network isolation, see: Openshift SDN.

3.8. 7. TROUBLESHOOTING

This section contains a list of common installation issues and provides guidance for their resolution.

- Previous deployment leaves dirty persistent volume claims
- Incorrectly pulling from the Docker registry
- Permissions issues for MySQL when persistent volumes are mounted locally
- Unable to upload logo or images
- Create secure routes on OpenShift
- APIcast on a Different Project from 3scale

3.8.1. Previous deployment leaves dirty persistent volume claims

Problem

A previous deployment attempt leaves a dirty Persistent Volume Claim (PVC) causing the MySQL container to fail to start.

Cause

Deleting a project in OpenShift does not clean the PVCs associated with it.

Solution

1. Find the PVC containing the erroneous MySQL data with the **oc get pvc** command:

```
# oc get pvc
NAME                 STATUS    VOLUME    CAPACITY   ACCESSMODES   AGE
backend-redis-storage   Bound     vol003    100Gi      RWO,RWX       4d
mysql-storage           Bound     vol006    100Gi      RWO,RWX       4d
system-redis-storage    Bound     vol008    100Gi      RWO,RWX       4d
system-storage          Bound     vol004    100Gi      RWO,RWX       4d
```

2. Stop the deployment of the system-mysql pod by clicking **cancel deployment** in the OpenShift UI.

3. Delete everything under the MySQL path to clean the volume.

4. Start a new **system-mysql** deployment.

3.8.2. Incorrectly pulling from the Docker registry

Problem

The following error occurs during installation:

```
svc/system-redis - 1EX.AMP.LE.IP:6379
  dc/system-redis deploys docker.io/rhscl/redis-32-rhel7:3.2-5.3
    deployment #1 failed 13 minutes ago: config change
```

Cause

OpenShift searches for and pulls container images by issuing the **docker** command. This command refers to the **docker.io** Docker registry instead of the **registry.access.redhat.com** Red Hat container registry.

This occurs when the system contains an unexpected version of the Docker containerized environment.

Solution

Use the appropriate version of the Docker containerized environment.

3.8.3. Permissions issues for MySQL when persistent volumes are mounted locally

Problem

The system-msql pod crashes and does not deploy causing other systems dependant on it to fail deployment. The pod log displays the following error:

```
[ERROR] Cannot start server : on unix socket: Permission denied
[ERROR] Do you already have another mysqld server running on socket: /var/lib/mysql/mysql.sock ?
[ERROR] Aborting
```

Cause

The MySQL process is started with inappropriate user permissions.

Solution

1. The directories used for the persistent volumes MUST have the write permissions for the root group. Having rw permissions for the root user is not enough as the MySQL service runs as a different user in the root group. Execute the following command as the root user:

   ```
   chmod -R g+w /path/for/pvs
   ```

2. Execute the following command to prevent SElinux from blocking access:

   ```
   chcon -Rt svirt_sandbox_file_t /path/for/pvs
   ```

3.8.4. Unable to upload logo or images

Problem

Unable to upload a logo - **system-app** logs display the following error:

```
Errno::EACCES (Permission denied @ dir_s_mkdir - /opt/system/public//system/provider-name/2
```

Cause

Persistent volumes are not writable by OpenShift.

Solution

Ensure your persistent volume is writable by OpenShift. It should be owned by root group and be group writable.

3.8.5. Create secure routes on OpenShift

Problem

Test calls do not work after creation of a new service and routes on OpenShift. Direct calls via curl also fail, stating: **service not available**.

Cause

3scale requires HTTPS routes by default, and OpenShift routes are not secured.

Solution

Ensure the **secure route** checkbox is clicked in your OpenShift router settings.

3.8.6. APIcast on a different project from 3scale

Problem

APIcast deploy fails (pod does not turn blue). The following error appears in the logs:

```
update acceptor rejected apicast-3: pods for deployment "apicast-3" took longer than 600 seconds to become ready
```

The following error appears in the pod:

```
Error synching pod, skipping: failed to "StartContainer" for "apicast" with RunContainerError: "GenerateRunContainerOptions: secrets \"apicast-configuration-url-secret\" not found"
```

Cause

The secret was not properly set up.

Solution

When creating a secret with APIcast v3, specify **apicast-configuration-url-secret**:

```
oc secret new-basicauth apicast-configuration-url-secret  --password=https://<ACCESS_TOKEN>@<TENANT_NAME>-admin.<WILDCARD_DOMAIN>
```

CHAPTER 4. 3SCALE API MANAGEMENT ON-PREMISES OPERATIONS AND SCALING GUIDE

4.1. INTRODUCTION

This document describes operations and scaling tasks of a Red Hat 3scale AMP 2.4 On-Premises installation.

4.1.1. Prerequisites

An installed and initially configured AMP On-Premises instance on a supported OpenShift version.

This document is not intended for local installations on laptops or similar end user equipment.

4.1.2. Further Reading

- Health and Liveness Monitoring
- OpenShift Documentation

4.2. RE-DEPLOYING APICAST

After you have deployed AMP On-Premises and your chosen APIcast deployment method, you can test and promote system changes through your AMP dashboard. By default, APIcast deployments on OpenShift, both built-in and on other OpenShift clusters, are configured to allow you to publish changes to your staging and production gateways through the AMP UI.

Redeploy APIcast on OpenShift:

1. Make system changes.
2. In the UI, deploy to staging and test.
3. In the UI, promote to production.

By default, APIcast retrieves and publishes the promoted update once every 5 minutes.

If you are using APIcast on the Docker containerized environment or a native installation, you must configure your staging and production gateways, and configure how often your gateway retrieves published changes. After you have configured your APIcast gateways, you can redeploy APIcast through the AMP UI.

To redeploy APIcast on the Docker containerized environment or a native installations:

1. Configure your APIcast gateway and connect it to AMP On-Premises.
2. Make system changes.
3. In the UI, deploy to staging and test.
4. In the UI, promote to production.

APIcast retrieves and publishes the promoted update at the configured frequency.

4.3. APICAST BUILT-IN WILDCARD ROUTING

The built-in APIcast gateways that accompany your on-preimses AMP deployment support wildcard domain routing at the subdomain level. This feature allows you to name a portion of your subdomain for your production and staging gateway public base URLs. To use this feature, you must have enabled it during the on-premises installation.

NOTE

Ensure that you are using the OpenShift Container Platform version that supports Wildcard Routing. For information on the supported versions, see Supported Configurations.

The AMP does not provide DNS capabilities, so your specified public base URL must match the DNS configuration specified in the **WILDCARD_DOMAIN** parameter of the OpenShift cluster on which it was deployed.

4.3.1. Modify Wildcards

Perform the following steps to modify your wildcards:

1. Log in to your AMP.
2. Navigate to your API gateway settings page: **APIs** → your API → **Integration** → **edit APIcast configuration**
3. Modify the staging and production public base URLs with a string prefix of your choice, adhere to these requirements:
    - API endpoints must not begin with a numeric character

The following is an example of a valid wildcard for a staging gateway on the domain **example.com**:

```
apiname-staging.example.com
```

More Information

For information on routing, see the OpenShift documentation.

4.4. SCALING UP AMP ON PREMISES

4.4.1. Scaling up Storage

As your APIcast deployment grows, you may need to increase the amount of storage available. How you scale up storage depends on which type of file system you are using for your persistent storage.

If you are using a network file system (NFS), you can scale up your persistent volume using the **oc edit pv** command:

```
oc edit pv <pv_name>
```

If you are using any other storage method, you must scale up your persistent volume manually using one of the methods listes in the following sections.

4.4.1.1. Method 1: Backup and Swap Persistent Volumes

1. Back up the data on your existing persistent volume.
2. Create and attach a target persistent volume, scaled for your new size requirements.
3. Create a pre-bound persistent volume claim, specify: The size of your new PVC The persistent volume name using the **volumeName** field.
4. Restore data from your backup onto your newly created PV.
5. Modify your deployment configuration with the name of your new PV:

```
oc edit dc/system-app
```

6. Verify your new PV is configured and working correctly.
7. Delete your previous PVC to release its claimed resources.

4.4.1.2. Method 2: Back up and Redeploy AMP

1. Back up the data on your existing persistent volume.
2. Shut down your 3scale pods.
3. Create and attach a target persistent volume, scaled for your new size requirements.
4. Restore data from your backup onto your newly created PV.
5. Create a pre-bound persistent volume claim. Specify:
   a. The size of your new PVC
   b. The persistent volume name using the **volumeName** field.
6. Deploy your AMP.yml.
7. Verify your new PV is configured and working correctly.
8. Delete your previous PVC to release its claimed resources.

4.4.2. Scaling up Performance

4.4.2.1. Configuring 3scale On-Premises Deployments

By default, 3scale deployments run one process per pod. You can increase performance by running more processes per pod. Red Hat recommends running 1-2 processes per core on each node.

Perform the following steps to add more processes to a pod:

1. Log in to your OpenShift cluster.

```
oc login
```

2. Switch to your 3scale project.

```
oc project <project_name>
```

3. Set the appropriate environment variable to the desired number of processes per pod.

    a. **APICAST_WORKERS** for APIcast pods (Red Hat recommends to keep this environment variable unset to allow APIcast to determine the number of workers by the number of CPUs available to the APIcast pod)

    b. **PUMA_WORKERS** for backend pods

    c. **UNICORN_WORKERS** for system pods

```
oc env dc/apicast --overwrite APICAST_WORKERS=<number_of_processes>
```

```
oc env dc/backend --overwrite PUMA_WORKERS=<number_of_processes>
```

```
oc env dc/system-app --overwrite UNICORN_WORKERS=<number_of_processes>
```

4.4.2.2. Vertical and Horizontal Hardware Scaling

You can increase the performance of your AMP deployment on OpenShift by adding resources. You can add more compute nodes as pods to your OpenShift cluster (horizontal scaling) or you can allocate more resources to existing compute nodes (vertical scaling).

Horizontal Scaling

You can add more compute nodes as pods to your OpenShift. If the additional compute nodes match the existing nodes in your cluster, you do not have to reconfigure any environment variables.

Vertical Scaling

You can allocate more resources to existing compute nodes. If you allocate more resources, you must add additional processes to your pods to increase performance.

NOTE

Red Hat does not recommend mixing compute nodes of a different specification or configuration on your 3scale deployment.

4.4.2.3. Scaling Up Routers

As your traffic increases, you must ensure your OCP routers can adequately handle requests. If your routers are limiting the throughput of your requests, you must scale up your router nodes.

4.4.2.4. Further Reading

- Scaling tasks, adding hardware compute nodes to OpenShift
- Adding Compute Nodes
- Routers

4.5. OPERATIONS TROUBLESHOOTING

4.5.1. Access Your Logs

Each component's deployment configuration contains logs for access and exceptions. If you encounter issues with your deployment, check these logs for details.

Follow these steps to access logs in 3scale:

1. Find the ID of the pod you want logs for:

```
oc get pods
```

2. Enter **oc logs** and the ID of your chosen pod:

```
oc logs <pod>
```

The system pod has two containers, each with a separate log. To access a container's log, specify the **--container** parameter with the **system-provider** and **system-developer**:

```
oc logs <pod> --container=system-provider
oc logs <pod> --container=system-developer
```

4.5.2. Job Queues

Job Queues contain logs of information sent from the **system-sidekiq** pods. Use these logs to check if your cluster is processing data. You can query the logs using the OpenShift CLI:

```
oc get jobs
```

```
oc logs <job>
```

CHAPTER 5. 3SCALE API MANAGEMENT HIGH AVAILABILITY AND EVALUATION

5.1. INTRODUCTION

This document describes the templates for High Availability and Evaluation used by Red Hat 3scale API Management 2.4 On-Premises installation.

5.2. PREREQUISITES

- You need to have an available OpenShift cluster to deploy elements of the High Availability and Evaluation templates.

5.3. HIGH AVAILABILITY TEMPLATE

The High Availability (HA) template allows you to have a HA setting for critical databases.

5.3.1. Prerequisites

- Before deploying the HA template, you must deploy and configure the external databases, and configure them in a HA configuration with a load-balanced endpoint.

5.3.2. Using the HA template

For HA, the template named **amp-ha-tech-preview.yml** allows you to deploy critical databases externally to OpenShift. This excludes:

- Memcached
- Sphinx
- Zync

Differences between the standard **amp.yml** template and **amp-ha-tech-preview.yml** include:

- Removal of the following elements:
  - backend-redis and its related components
  - system-redis and its related components
  - system-mysql and its related components
  - Redis and MySQL related ConfigMaps
  - MYSQL_IMAGE, REDIS_IMAGE, MYSQL_USER, MYSQL_ROOT_PASSWORD parameters
- By default, increased from 1 to 2 the number of replicas for non-database **DeploymentConfig** object types.
- Addition of the following mandatory parameters, allowing you the control of the location of external databases:
  - BACKEND_REDIS_STORAGE_ENDPOINT

- BACKEND_REDIS_QUEUES_ENDPOINT
- SYSTEM_REDIS_URL
- APICAST_STAGING_REDIS_URL
- APICAST_PRODUCTION_REDIS_URL
- SYSTEM_DATABASE_URL

With **amp-ha-tech-preview.yml**, you need to configure database connections (excluding **system-memcache**, **zync-database** and **system-sphinx** that do not contain permanent data) out of the cluster via the newly added mandatory parameters. The endpoints require database load-balanced connection strings, including authentication information. Also, for the non-database deployments, the number of pod replicas is increased to 2 by default to have redundancy at application-level.

5.4. EVALUATION TEMPLATE

For evaluation purposes, there is a template named **amp-eval-tech-preview.yml** that deploys a 3scale environment without resource requests nor limits.

The only functional difference compared to the standard **amp.yml** template is that the resource limits and requests have been removed. This means that in this version the minimum hardware requirements have been removed on the pods at CPU and Memory level. This template is intended only for evaluation, testing, and development purposes as it tries to deploy the components in a best-effort way with the given hardware resources.

CHAPTER 6. REDIS HIGH AVAILABILITY (HA) SUPPORT FOR 3SCALE

NOTE

There are known issues with Redis high availability (HA) support for 3scale. For more information, see the Red Hat 3scale API Management 2.4 release notes, Section 1.5. Known Issues in the release notes.

6.1. INTRODUCTION

High availability (HA) is provided for most components by the OpenShift Container Platform (OCP). For more information see OpenShift Container Platform 3.11 Chapter 30. High Availability .

The database components for HA in 3scale include:

- **system-redis**: provides temporary storage for background jobs for 3scale API Management and it is also used as a message bus for *Ruby* processes of **system-app** pods.
- **backend-redis**: used for statistics storage and temporary job storage.

NOTE

Both **system-redis** and **backend-redis** can be replaced by the *Redis Cluster* (open-source or Redis Labs).

The following **env vars** can be set into **system-**(**app**,**sidekiq**,**sphinx**) deployment configurations, though it is only a requirement for **Redis Enterprise**:

- **MESSAGE_BUS_REDIS_URL** (a redis URL)
- **REDIS_NAMESPACE** (a short string to namespace Sidekiq's Redis keys)
- **MESSAGE_BUS_REDIS_NAMESPACE** (a short string to namespace System message bus's Redis keys)

A new pod is created automatically when the Redis pod dies or is killed by OCP and the data is restored from the persistent storage, so the pod continues to work. In the scenario described, there would be a small amount of downtime while the new pod is being started. This is due to the limitation that Redis does not support a multi-master setup. Downtime can be reduced by preloading the Redis images onto all nodes that have Redis deployed to them, which will speed up the pod restart.

6.2. SETTING UP REDIS FOR ZERO DOWNTIME

If zero downtime is required, Redis would need to be set up outside of OCP. There are several ways to set it up using the configuration options of 3scale pods:

- Set up your own self-managed Redis
- Use Redis Sentinel: Reference Redis Sentinel Documentation
- Redis provided as a service:
  For example by:

- Amazon ElastiCache
- Redis Labs

NOTE

Red Hat does not provide support for the above mentioned services. The mention of any such services does not imply endorsement by Red Hat of the products or services. You agree that Red Hat is not responsible or liable for any loss or expenses that may result due to your use of (or reliance on) any external content.

6.3. CONFIGURATING BACKEND COMPONENTS FOR 3SCALE

There are configuration settings in 3scale API Management 2.4 to configure Redis HA (failover) for the backend component. They can be set as environment variables in the following deployment configurations: **backend-cron**, **backend-listener**, and **backend-worker**:

- **CONFIG_REDIS_SENTINEL_HOSTS** and **CONFIG_QUEUES_SENTINEL_HOSTS**:
  A comma-separated list of Sentinel hosts for the main statistics database and the Resque background job database.

  NOTE

  Values should be in the format: name:value **<host>:<port>** For example: **host1:26379**, **host2:26379**, or **host3:26379**

- **CONFIG_REDIS_SENTINEL_ROLE** and **CONFIG_QUEUES_SENTINEL_ROLE**:
  The role of each Sentinels group, either *master* or *slave*. Currently only *master* (default) is supported.

This makes the value of **CONFIG_REDIS_PROXY** and **CONFIG_QUEUES_MASTER_NAME** take the meaning of Sentinel group name instead of a specific server.

- When no Sentinel hosts are configured, the environment variables **CONFIG_REDIS_PROXY** and **CONFIG_QUEUES_MASTER_NAME** can use URLs and support password-protected databases, for example:
  CONFIG_REDIS_PROXY=redis://user:password@server:port/database
  - The connection is then established with the password-protected Redis instance.
- When Sentinel hosts are configured, the password must be set in the Sentinel configuration and use the Sentinel group name instead: **CONFIG_REDIS_PROXY=master_group**

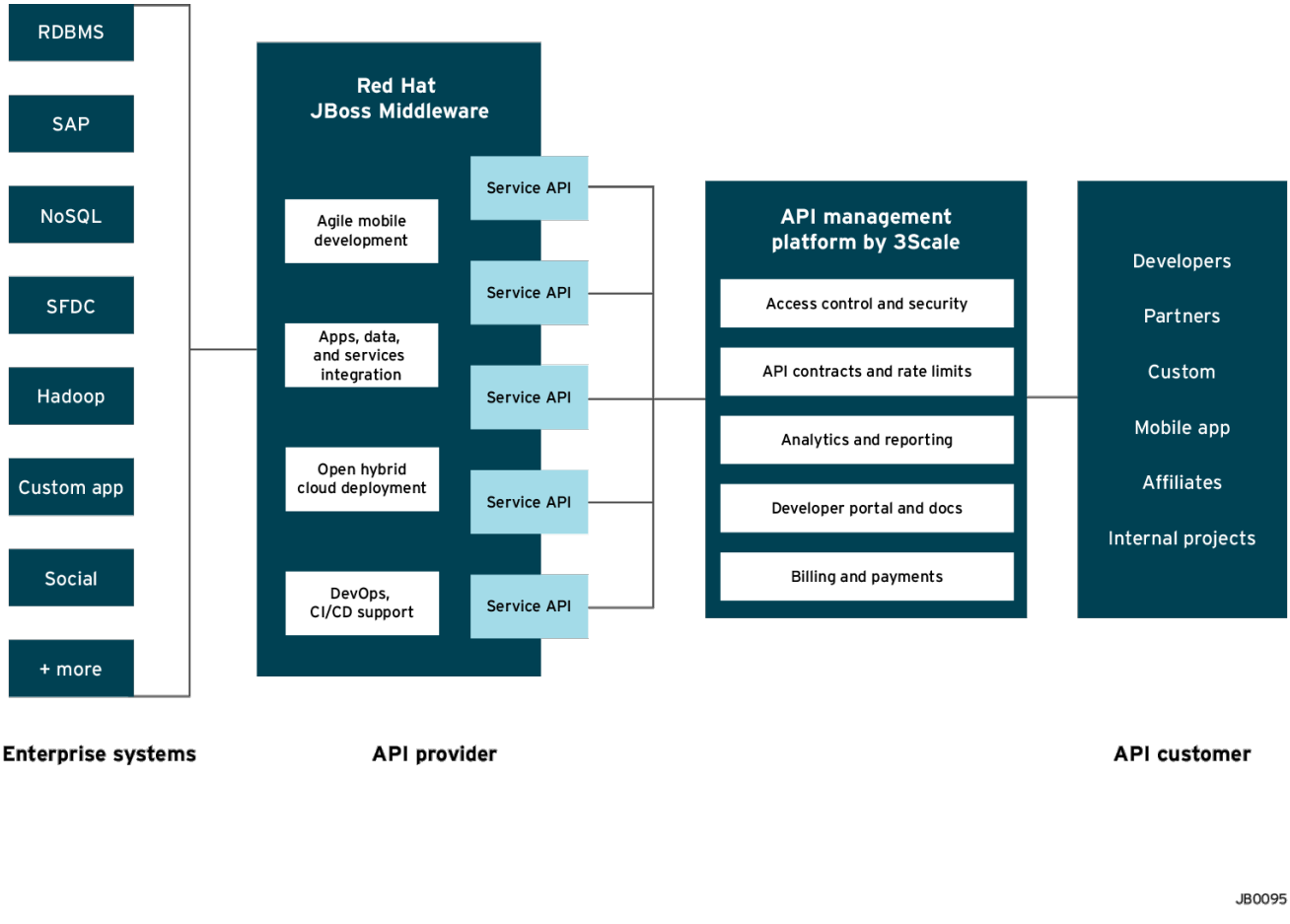
CHAPTER 7. HOW TO DEPLOY A FULL-STACK API SOLUTION WITH FUSE, 3SCALE, AND OPENSHIFT

This tutorial describes how to get a full-stack API solution (API design, development, hosting, access control, monetization, etc.) using Red Hat JBoss xPaaS for OpenShift and 3scale API Management Platform - Cloud.

The tutorial is based on a collaboration between Red Hat and 3scale to provide a full-stack API solution. This solution includes design, development, and hosting of your API on the Red Hat JBoss xPaaS for OpenShift, combined with the 3scale API Management Platform for full control, visibility, and monetization features.

The API itself can be deployed on Red Hat JBoss xPaaS for OpenShift, which can be hosted in the cloud as well as on premise (that's the Red Hat part). The API management (the 3scale part) can be hosted on Amazon Web Services (AWS), using 3scale APIcast or OpenShift. This gives a wide range of different configuration options for maximum deployment flexibility.

The diagram below summarizes the main elements of this joint solution. It shows the whole integration chain including enterprise backend systems, middleware, API management, and API customers.



For specific support questions, please contact support.

This tutorial shows three different deployment scenarios step by step:

1. Scenario 1 – A Fuse on OpenShift application containing the API. The API is managed by 3scale with the API gateway hosted on Amazon Web Services (AWS) using the 3scale AMI.

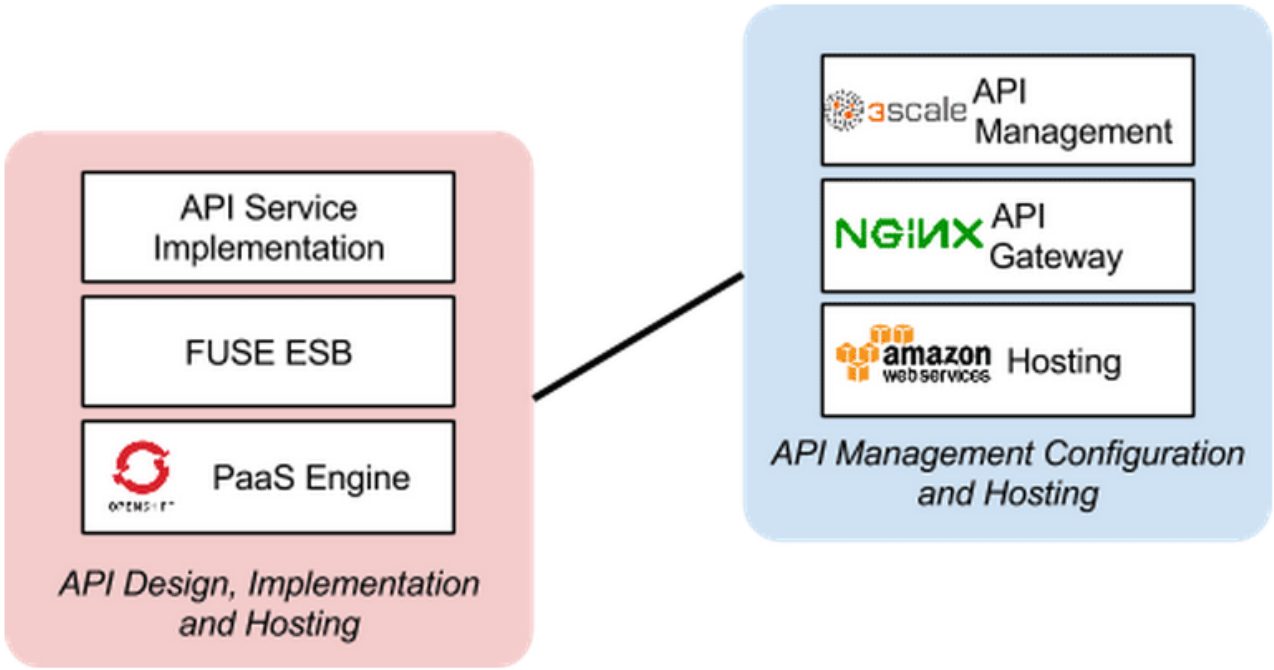
2. Scenario 2 – A Fuse on OpenShift application containing the API. The API is managed by 3scale with the API gateway hosted on APIcast (3scale's cloud hosted API gateway).

3. Scenario 3 – A Fuse on OpenShift application containing the API. The API is managed by 3scale with the API gateway hosted on OpenShift

This tutorial is split into four parts:

- Part 1: Fuse on OpenShift setup to design and implement the API
- Part 2: Configuration of 3scale API Management
- Part 3: Integration of your API services
- Part 4: Testing the API and API management

The diagram below shows the roles the various parts play in this configuration.

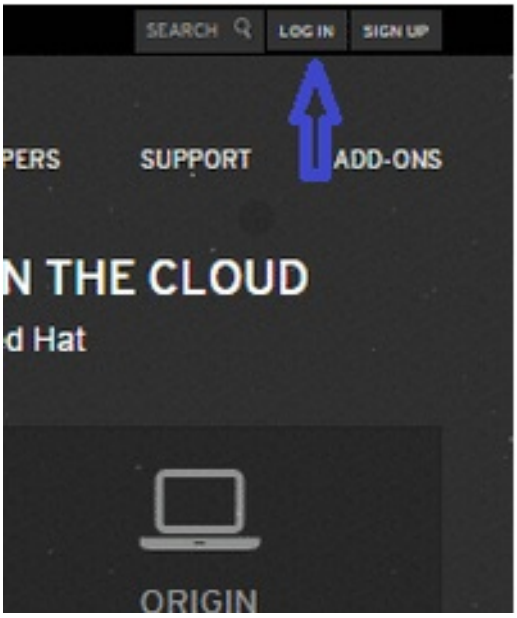


7.1. PART 1: FUSE ON OPENSHIFT SETUP

You will create a Fuse on OpenShift application that contains the API to be managed. You will use the REST quickstart that is included with Fuse 6.1. This requires a medium or large gear, as using the small gear will result in memory errors and/or horrible performance.

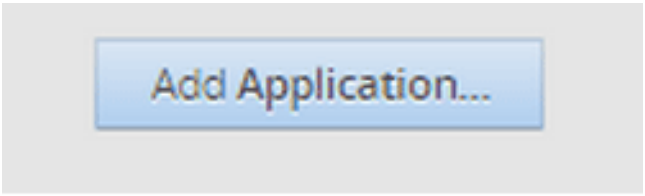
7.1.1. Step 1

Sign in to your OpenShift online account. Sign up for an OpenShift online account if you don't already have one.



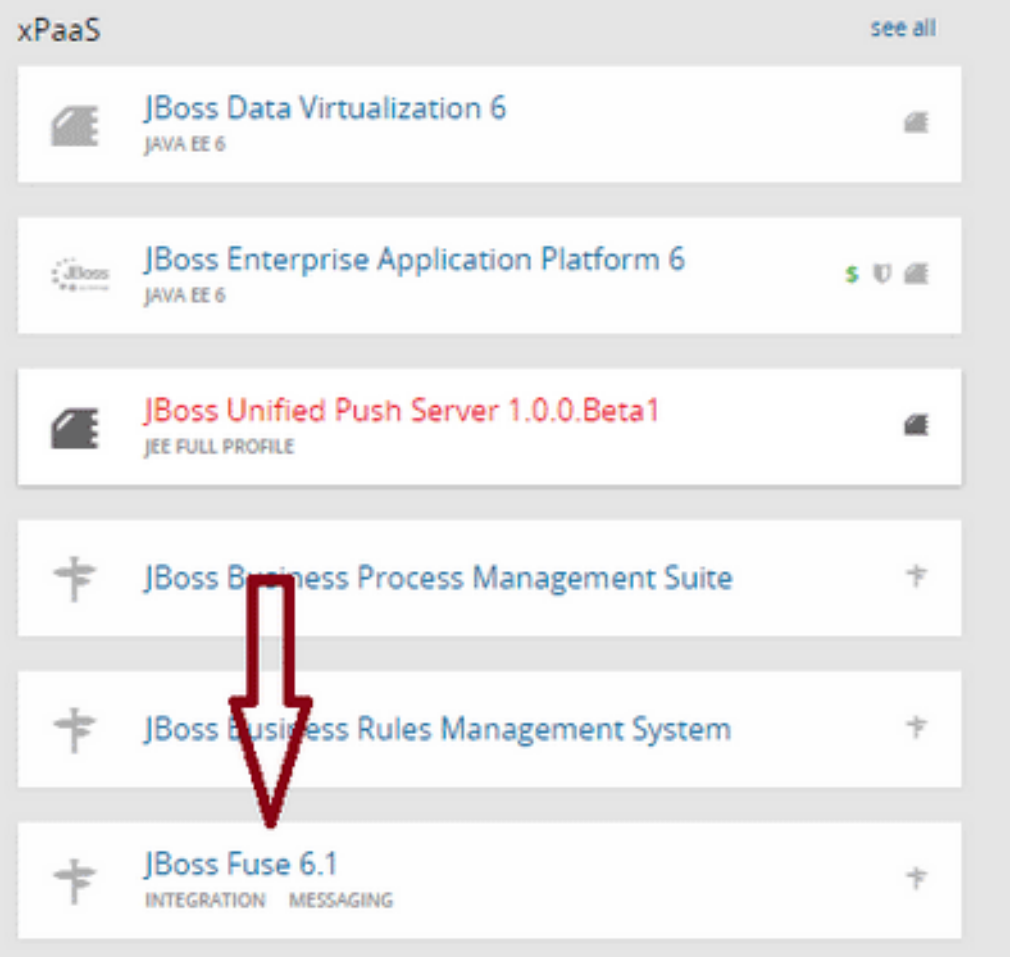
7.1.2. Step 2

Click the "add application" button after signing in.



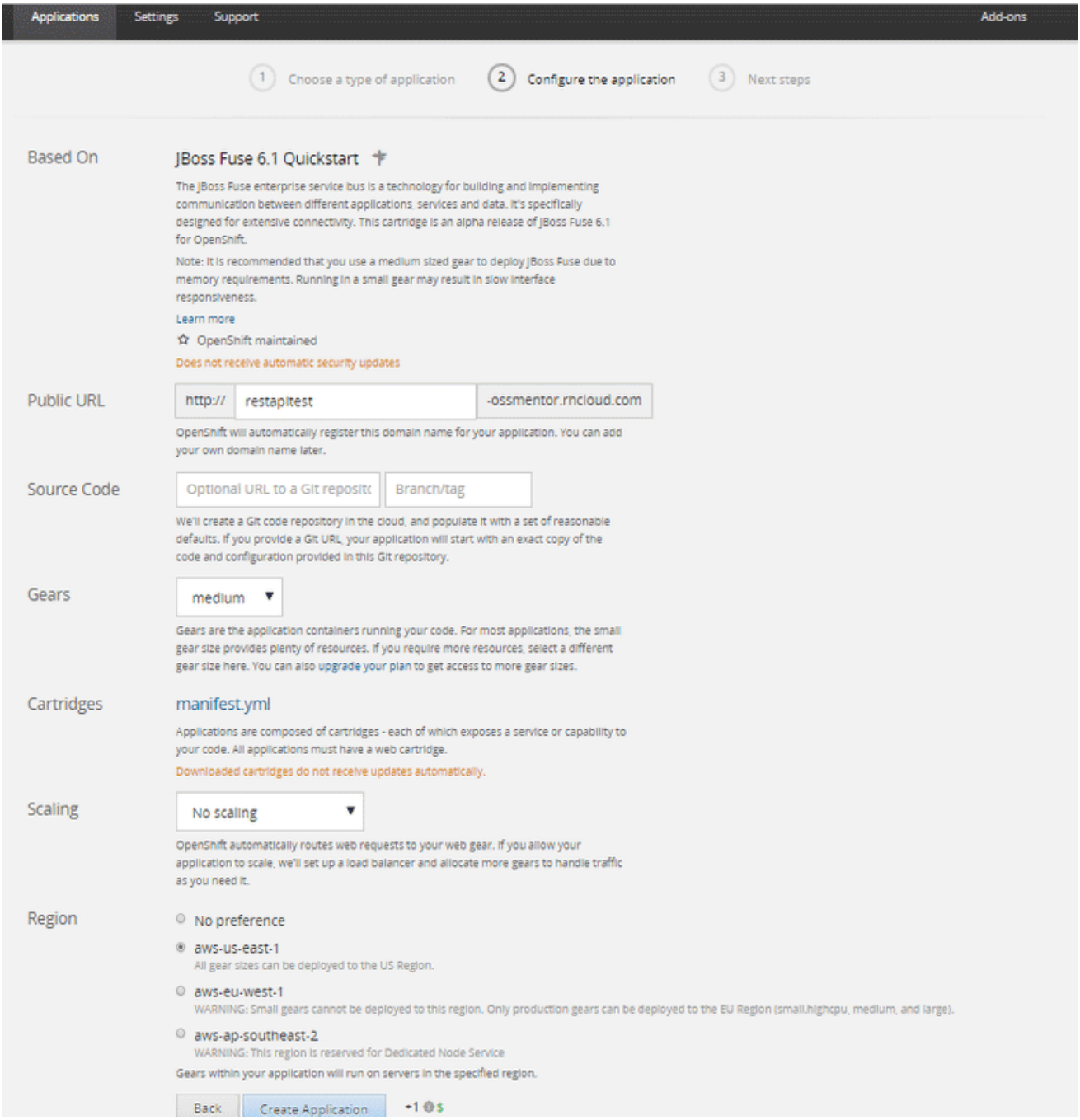
7.1.3. Step 3

Under xPaaS, select the Fuse type for the application.



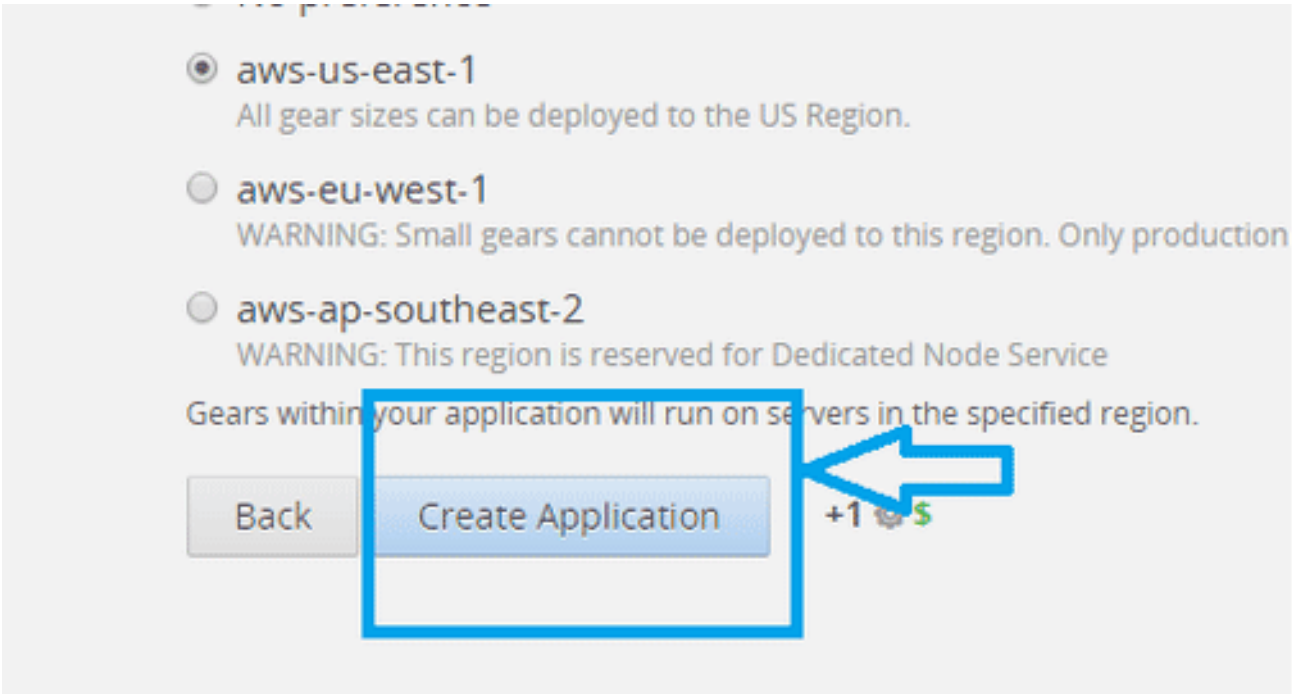
7.1.4. Step 4

Now configure the application. Enter the subdomain you'd like your application to show up under, such as "restapitest". This will give a full URL of the form "appname-domain.rhcloud.com" – in the example below "restapitest-ossmentor.rhcloud.com". Change the gear size to medium or large, which is required for the Fuse cartridge. Now click on "create application".



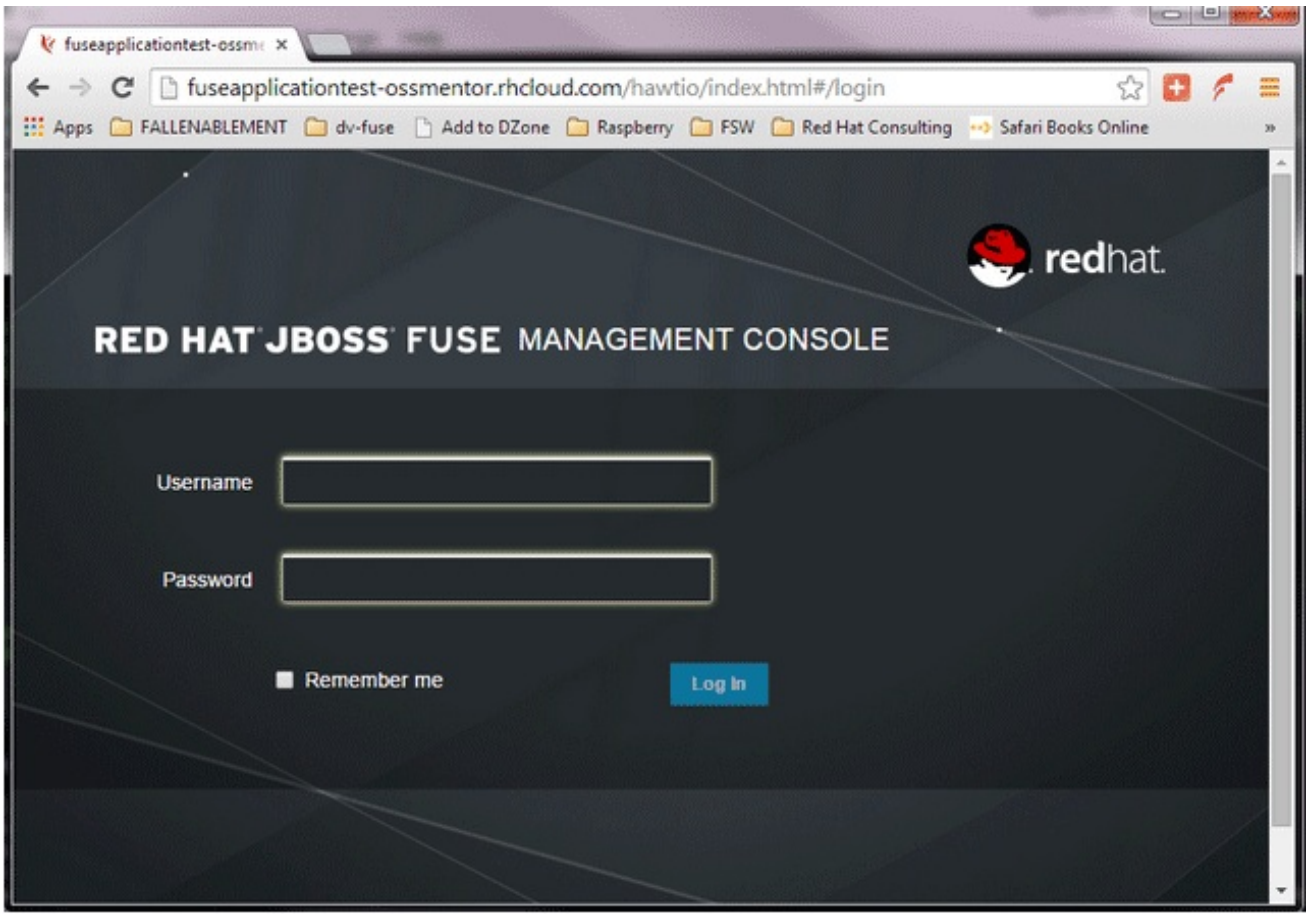
7.1.5. Step 5

Click "create application".



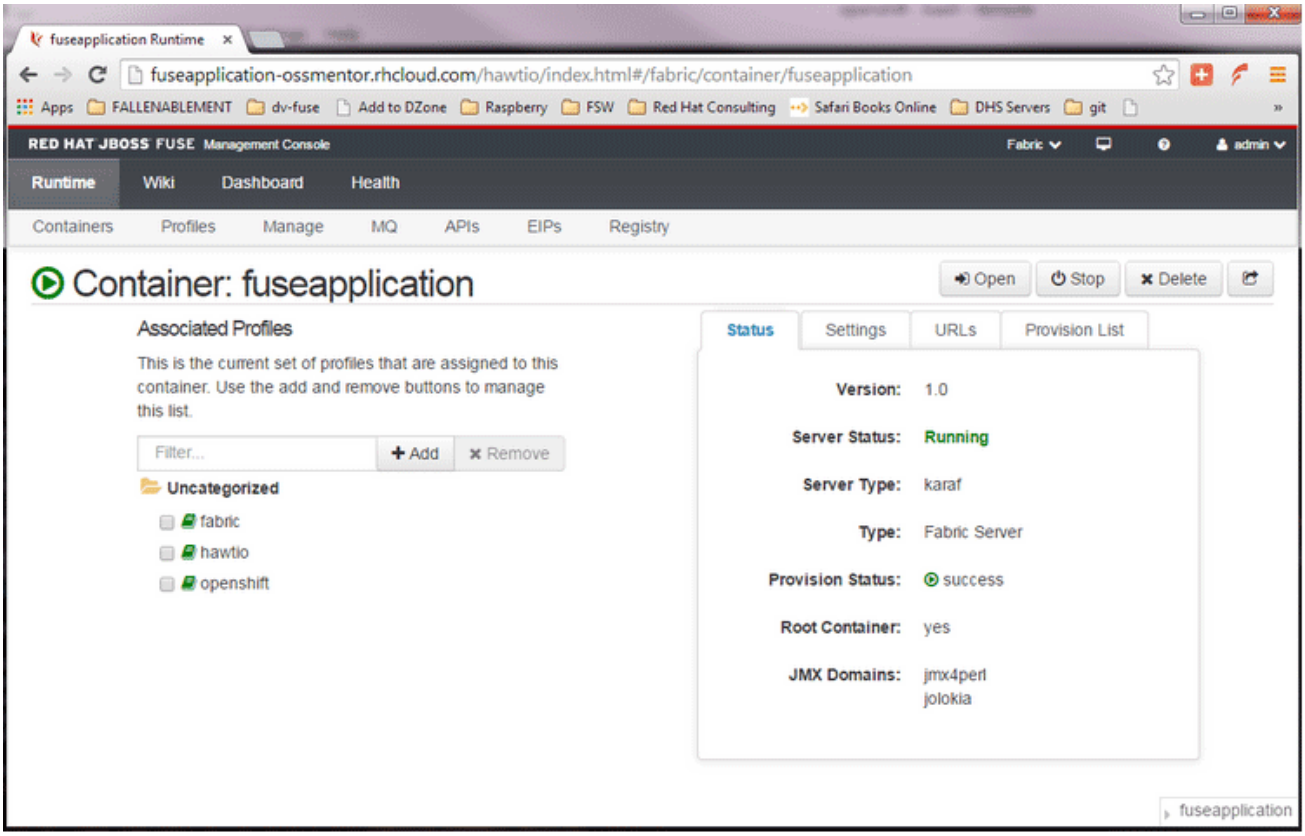
7.1.6. Step 6

Browse the application hawtio console and sign in.



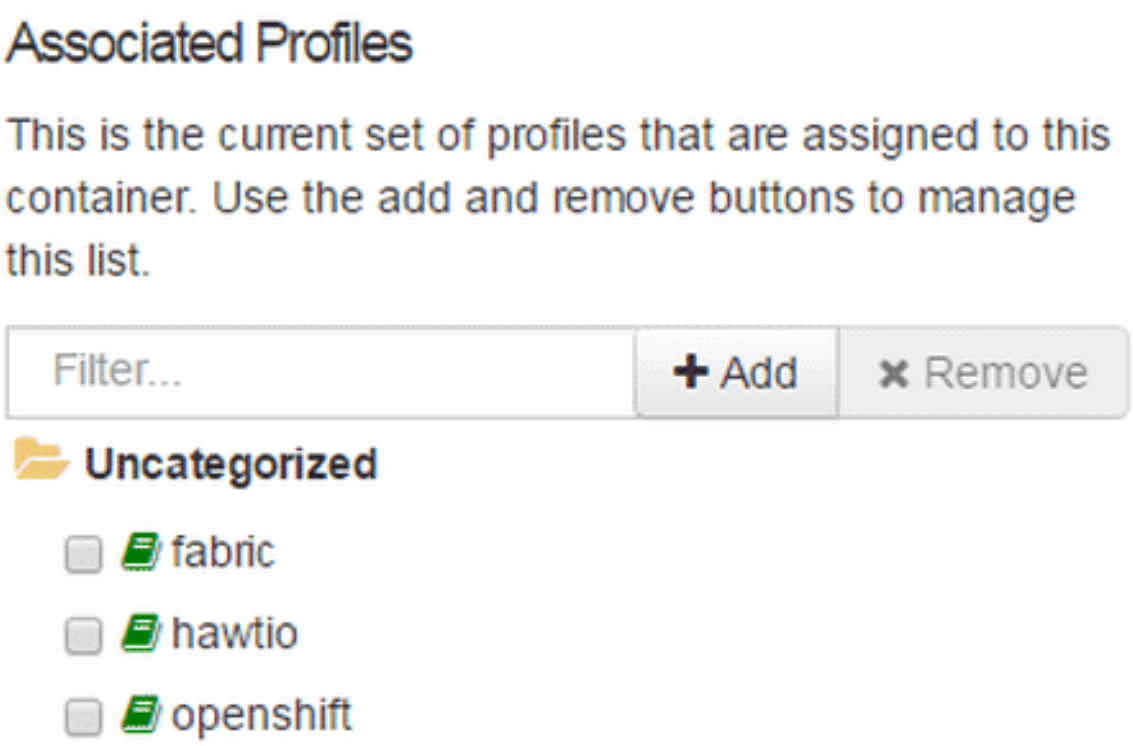
7.1.7. Step 7

After signing in, click on the "runtime" tab and the container, and add the REST API example.



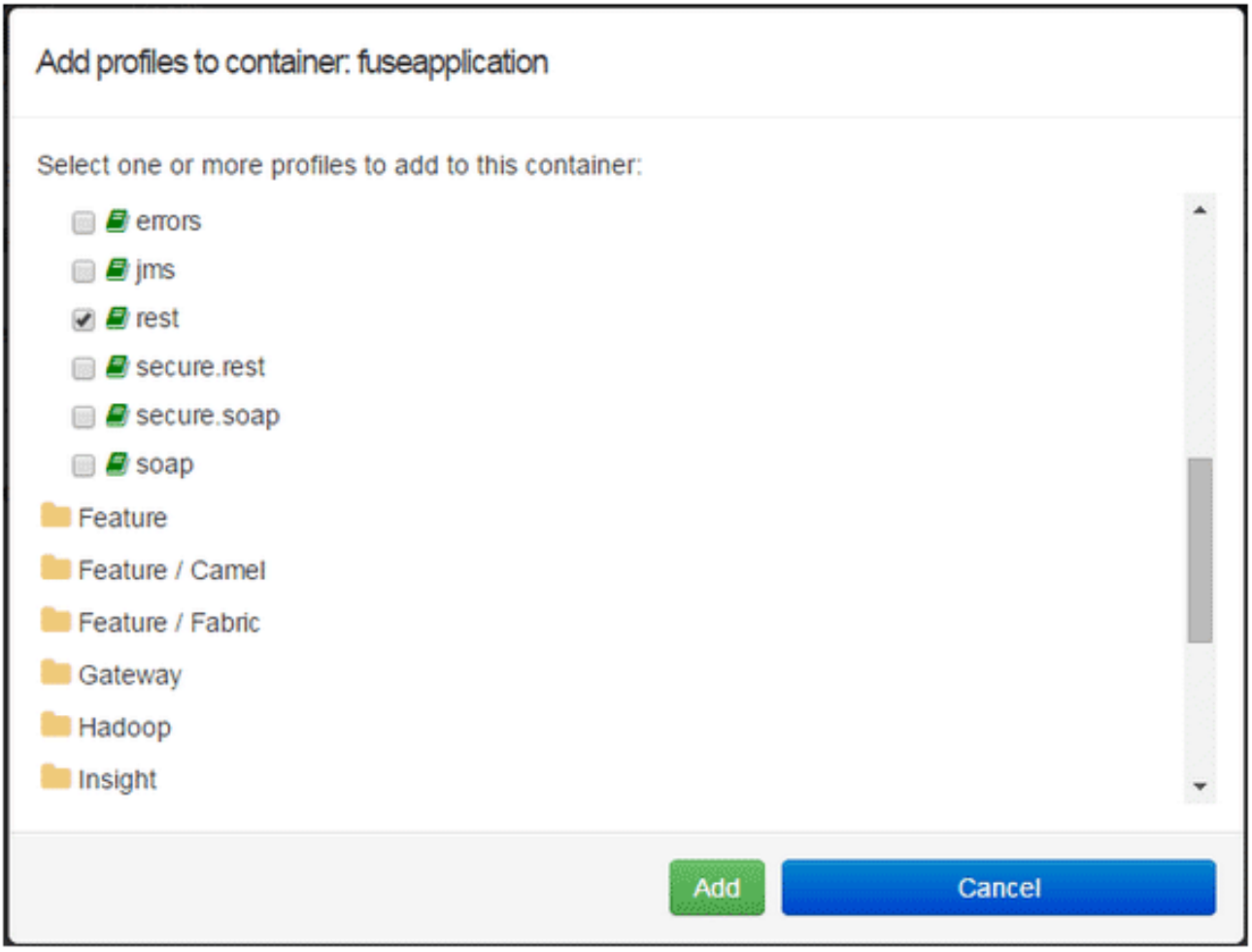
7.1.8. Step 8

Click on the "add a profile" button.



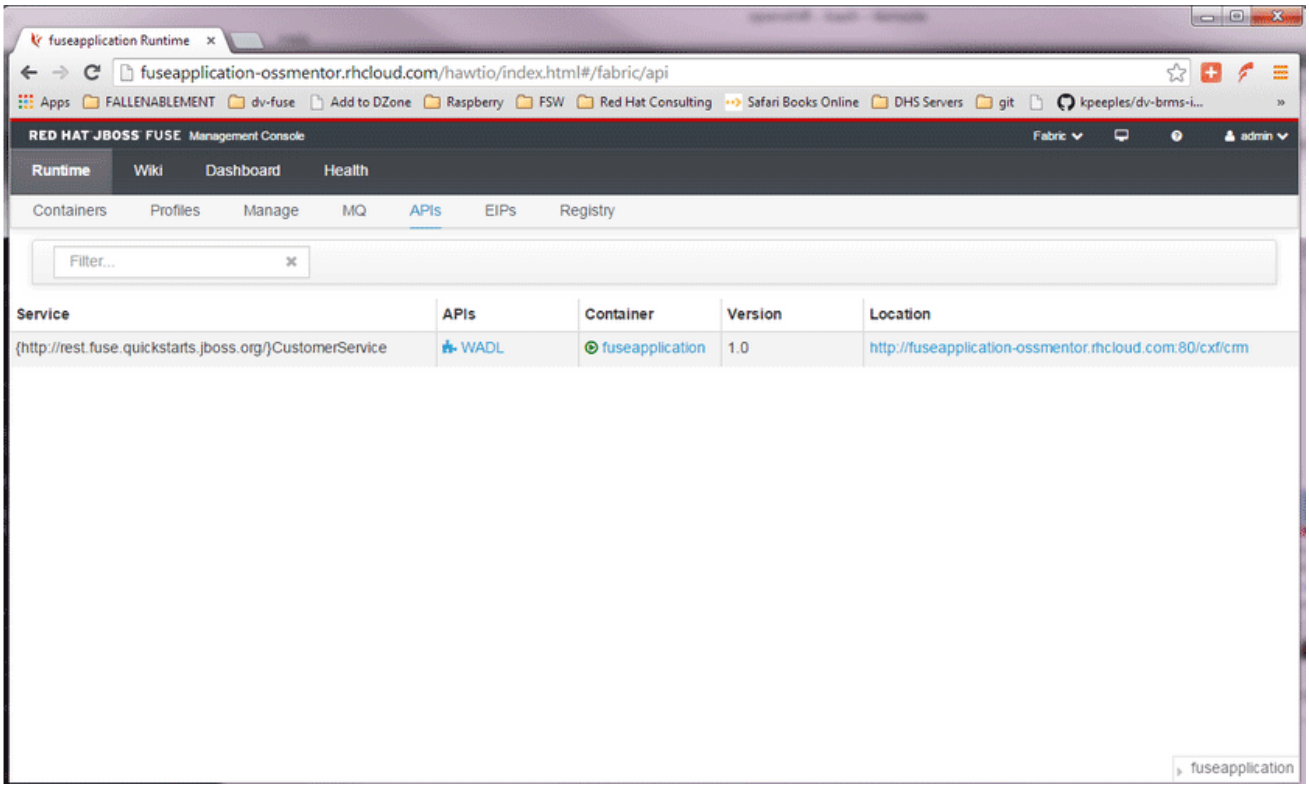
7.1.9. Step 9

Scroll down to examples/quickstarts and click the "REST" checkbox, then "add". The REST profile should show up on the container associated profile page.



7.1.10. Step 10

Click on the runtime/APIs tab to verify the REST API profile.



7.1.11. Step 11

Verify the REST API is working. Browse to customer 123, which will return the ID and name in XML format.

7.2. PART 2: CONFIGURE 3SCALE API MANAGEMENT

To protect the API that you just created in Part 1 using 3scale API Management, you first must conduct the according configuration, which is then later deployed according to one of the three scenarios presented.

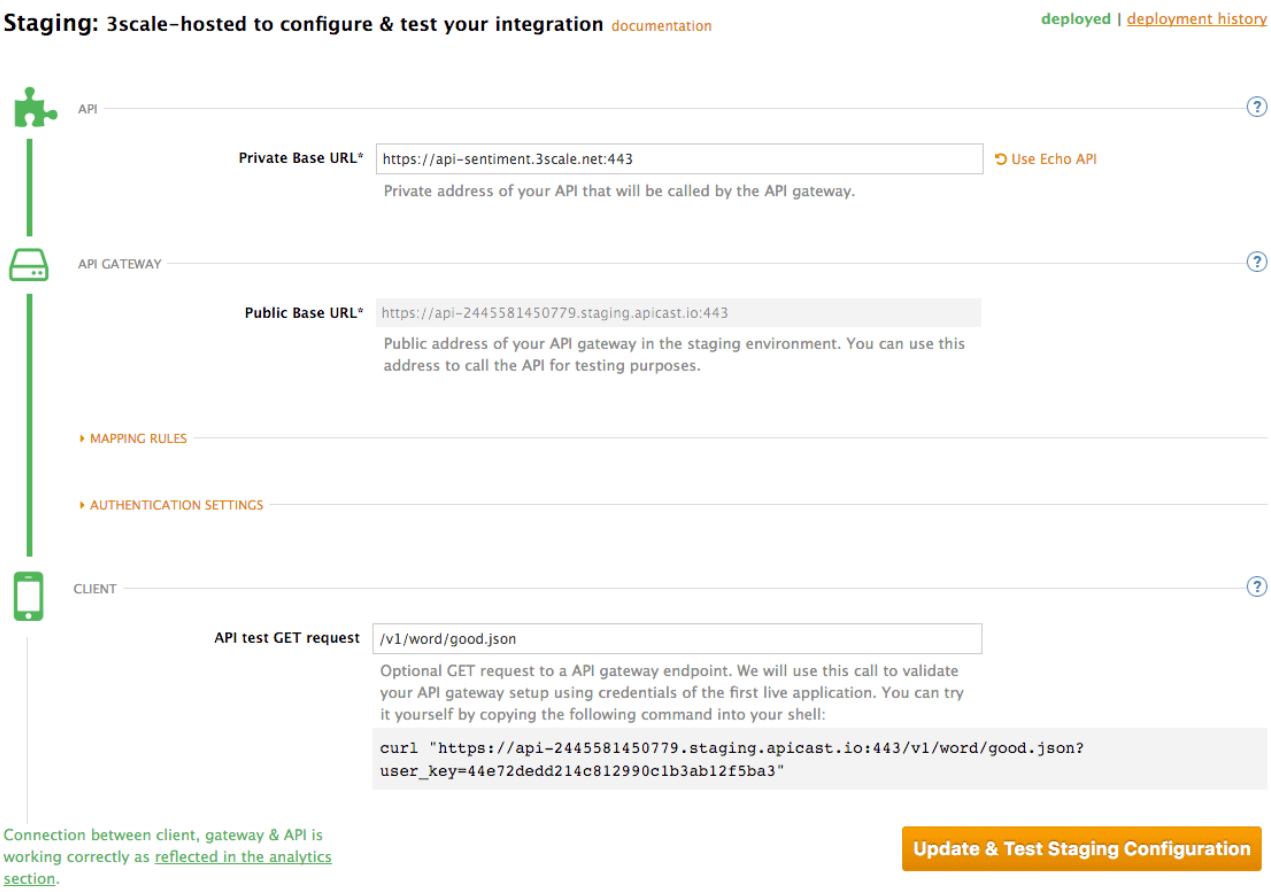
Once you have your API set up on OpenShift, you can start setting it up on 3scale to provide the management layer for access control and usage monitoring.

7.2.1. Step 1

Log in to your 3scale account. You can sign up for a 3scale account at www.3scale.net if you don't already have one. When you log in to your account for the first time, follow the wizard to learn the basics about integrating your API with 3scale.

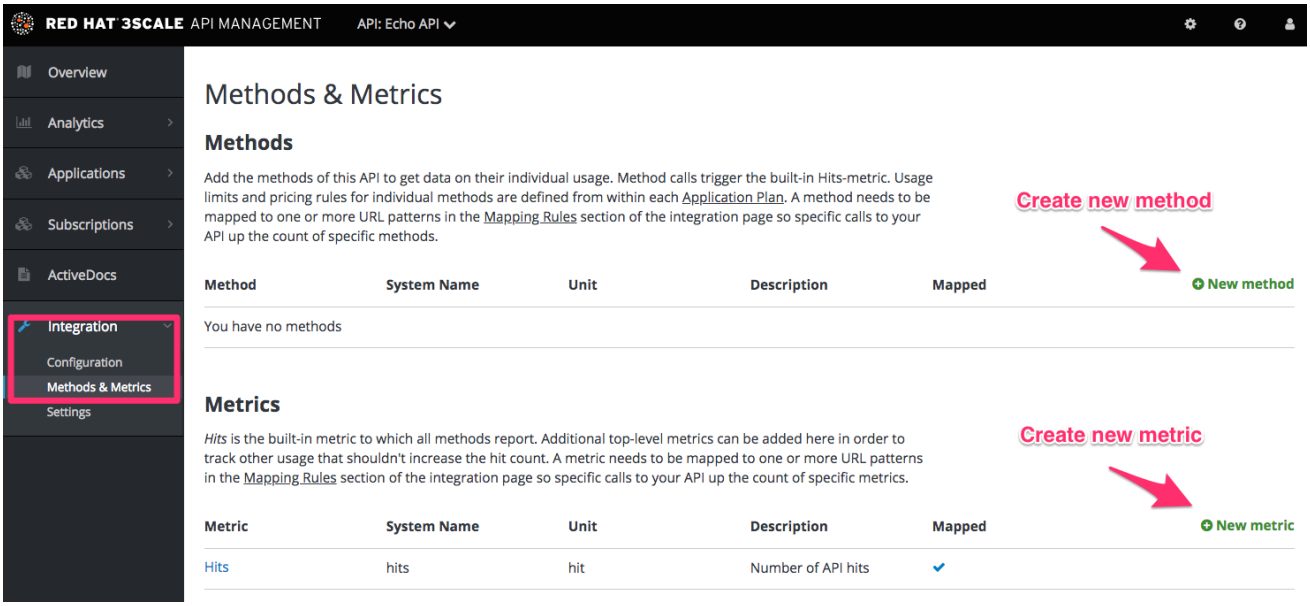
7.2.2. Step 2

In **[your_API_name] > Integration > Configuration** you can enter the public URL for the Fuse application on OpenShift that you just created, e.g. "restapitest-ossmentor.rhcloud.com" and click on **Test**. This will test your setup against the 3scale API Gateway in the staging environment. The staging API gateway allows you to test your 3scale setup before deploying your proxy configuration to AWS.



7.2.3. Step 3

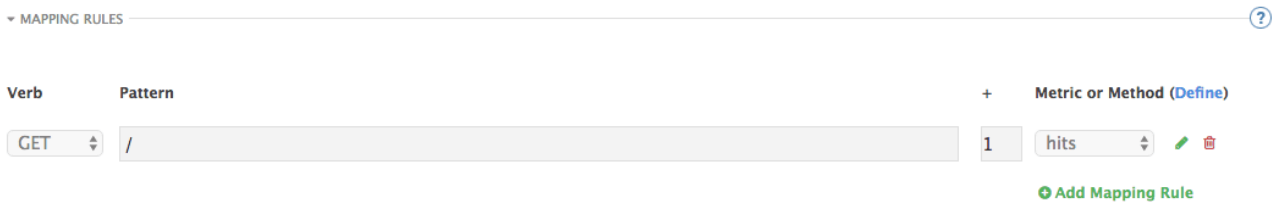
The next step is to set up the API methods that you want to monitor and rate limit. To do that go to **[your_API_name] > Integration > Methods & Metrics**and click on 'New method'.



For more details on creating methods, visit our API definition tutorial.

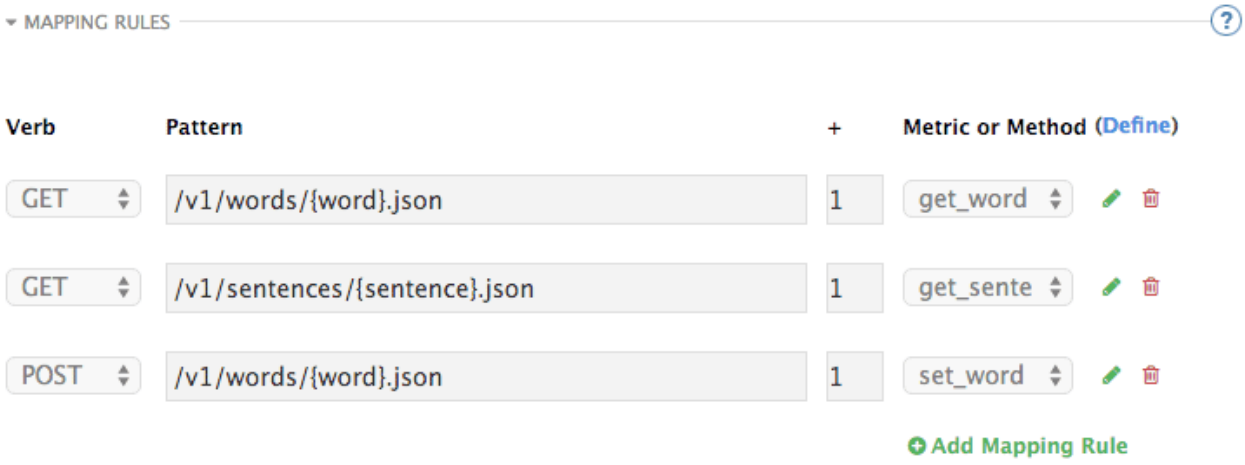
7.2.4. Step 4

Once you have all of the methods that you want to monitor and control set up under the application plan, you'll need to map these to actual HTTP methods on endpoints of your API. Go back to the integration page and expand the "mapping rules" section.



Create mapping rules for each of the methods you created under the application plan.

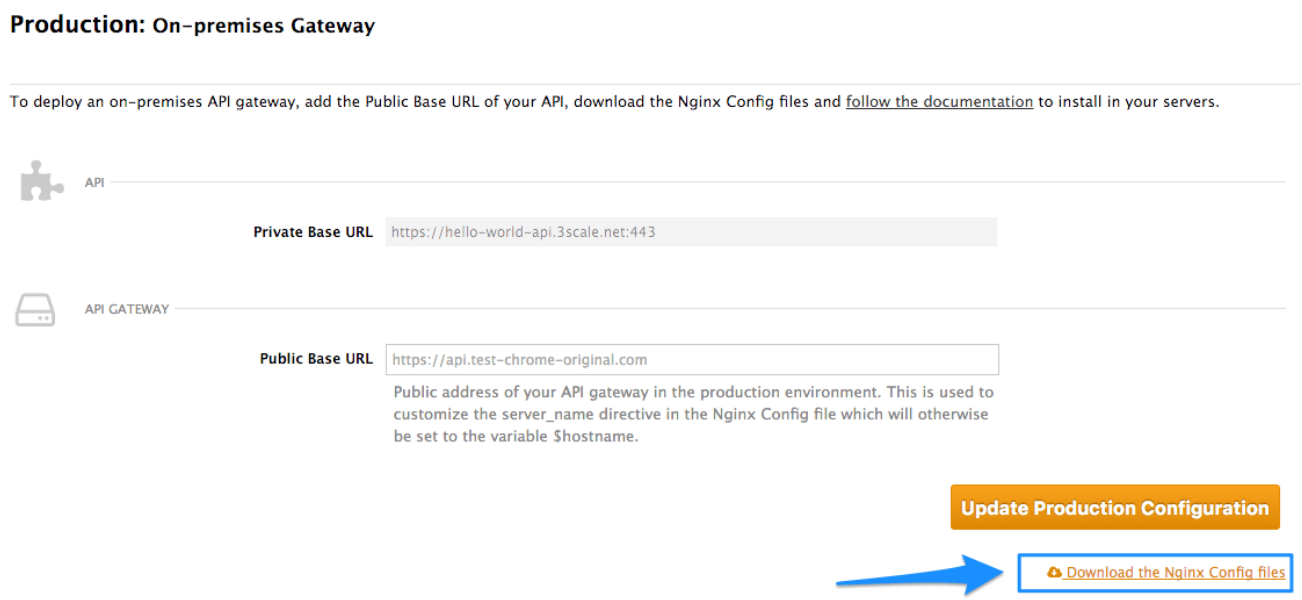
Once you have done that, your mapping rules will look something like this:



For more details on mapping rules, visit our tutorial about mapping rules.

7.2.5. Step 5

Once you've clicked "update and test" to save and test your configuration, you are ready to download the set of configuration files that will allow you to configure your API gateway on AWS. For the API gateway, you should use a high-performance, open-source proxy called nginx. You will find the necessary configuration files for nginx on the same integration page by scrolling down to the "production" section.



The next section will now take you through various hosting scenarios.

7.3. PART 3: INTEGRATION OF YOUR API SERVICES

There are several ways in which you can integrate your API services in 3scale. Choose the deployment option that best fits your needs.

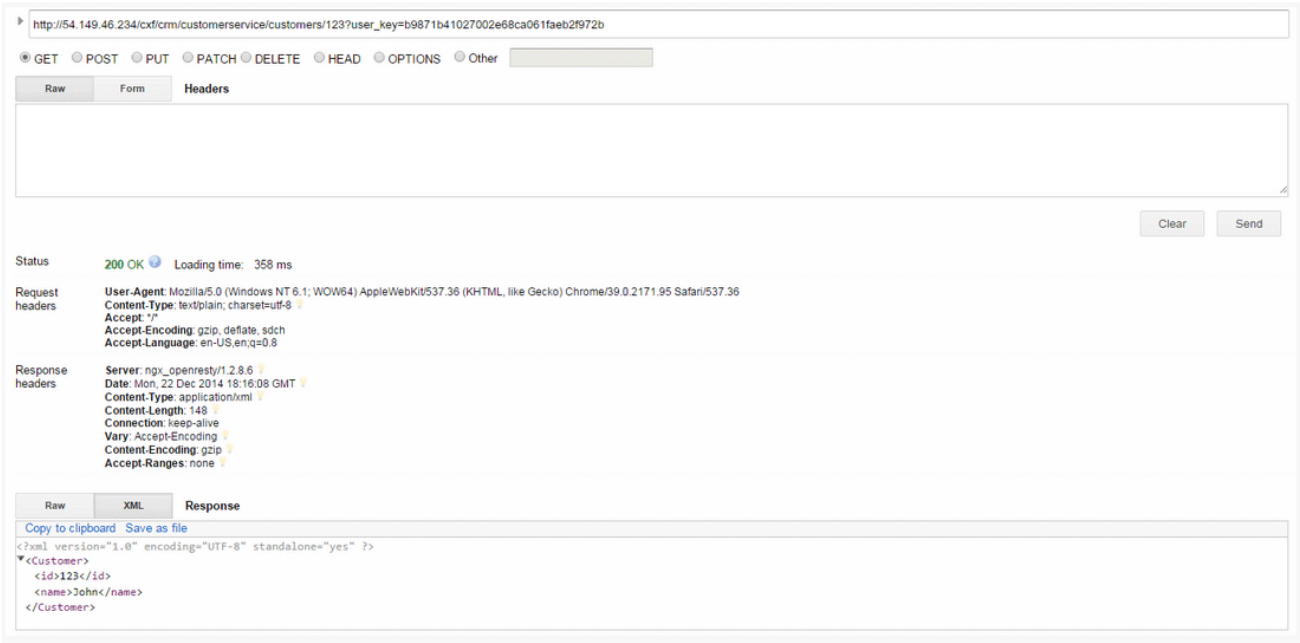
7.4. PART 4: TESTING THE API AND API MANAGEMENT

Testing the correct functioning of the API and the API Management is independent from the chosen scenario. You can use your favorite REST client and run the following commands.

7.4.1. Step 1

Retrieve the customer instance with id 123.

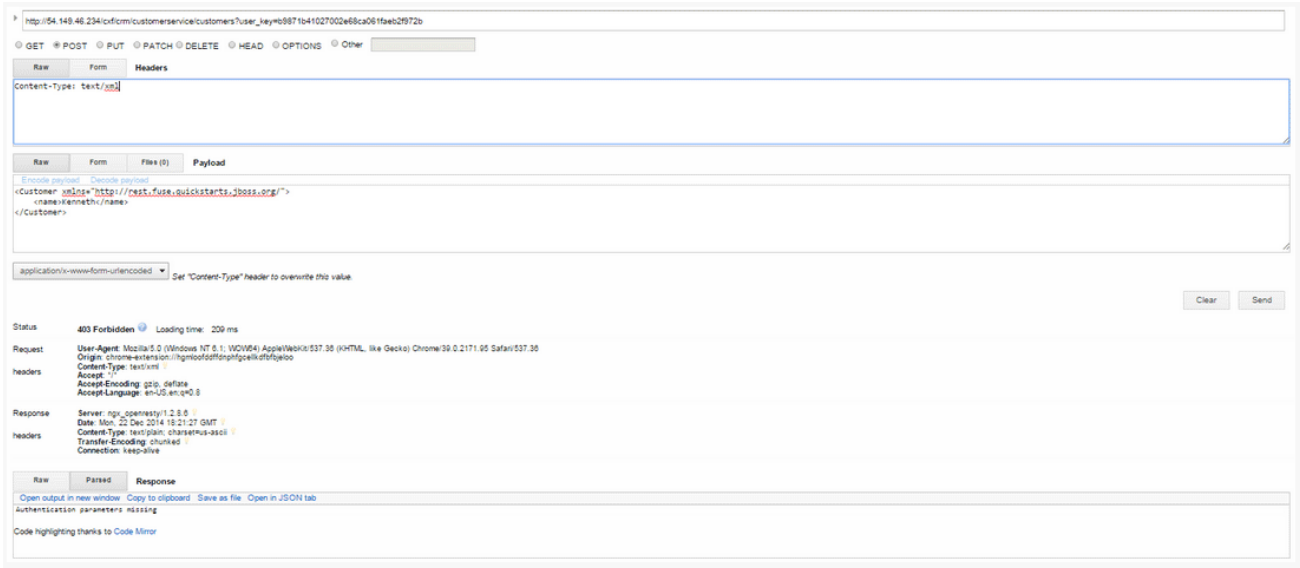
```
http://54.149.46.234/cxf/crm/customerservice/customers/123?
user_key=b9871b41027002e68ca061faeb2f972b
```



7.4.2. Step 2

Create a customer.

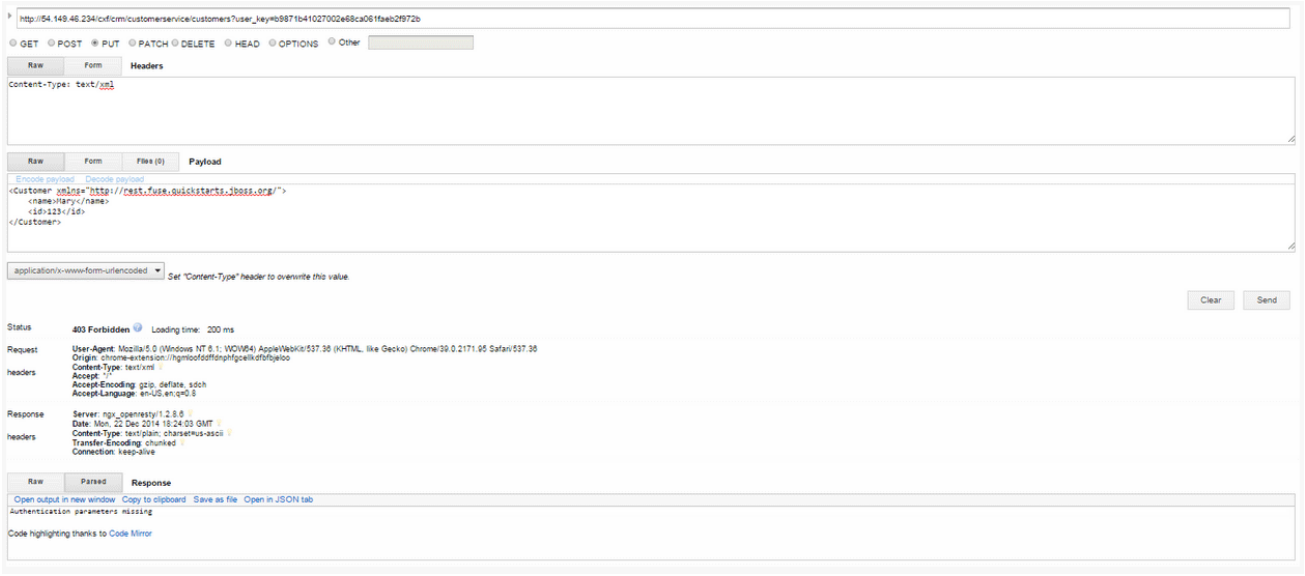
> http://54.149.46.234/cxf/crm/customerservice/customers?
> user_key=b9871b41027002e68ca061faeb2f972b



7.4.3. Step 3

Update the customer instance with id 123.

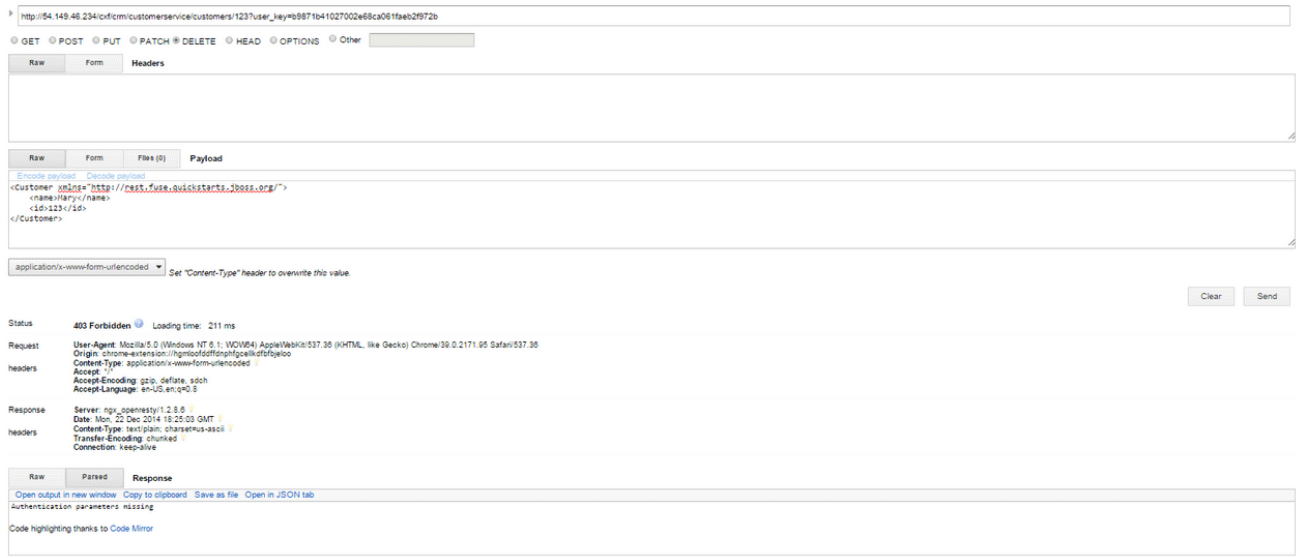
> http://54.149.46.234/cxf/crm/customerservice/customers?
> user_key=b9871b41027002e68ca061faeb2f972b



7.4.4. Step 4

Delete the customer instance with id 123.

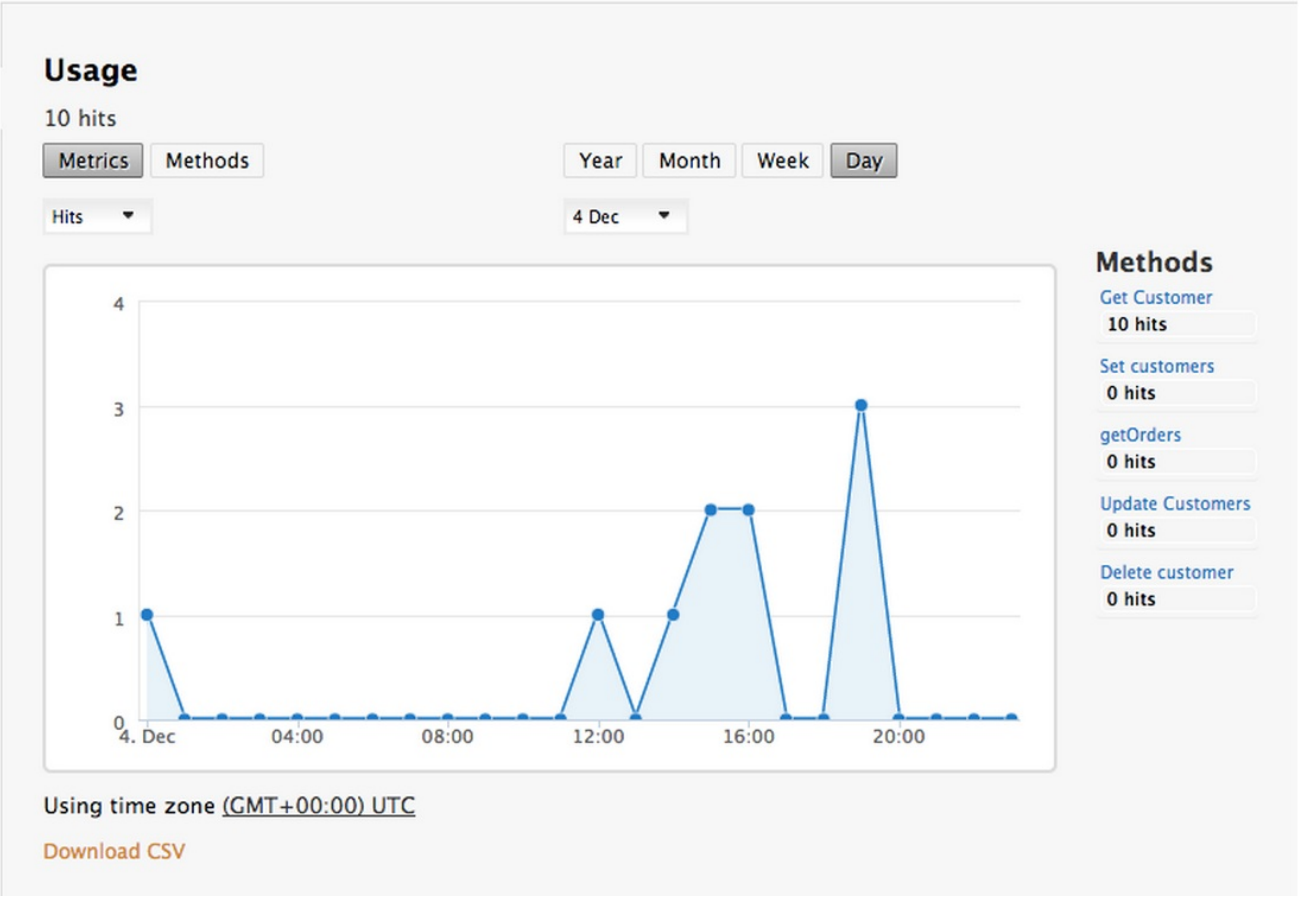
> http://54.149.46.234/cxf/crm/customerservice/customers/123?
> user_key=b9871b41027002e68ca061faeb2f972b



7.4.5. Step 5

Check the API Management analytics of your API.

If you now log back in to your 3scale account and go to Monitoring > Usage, you can see the various hits of the API endpoints represented as graphs.



This is just one element of API Management that brings you full visibility and control over your API. Other features include:

1. Access control
2. Usage policies and rate limits
3. Reporting
4. API documentation and developer portals
5. Monetization and billing

For more details about the specific API Management features and their benefits, please refer to the 3scale API Management Platform product description.

For more details about the specific Red Hat JBoss Fuse product features and their benefits, please refer to the JBOSS FUSE Overview.

For more details about running Red Hat JBoss Fuse on OpenShift, please refer to the Getting Started with JBoss Fuse on OpenShift.