Red Hat OpenShift Service on AWS 4

Prepare your environment

Planning, limits, and scalability for Red Hat OpenShift Service on AWS

Last Updated: 2022-04-29
Planning, limits, and scalability for Red Hat OpenShift Service on AWS
Abstract

This document provides planning considerations for Red Hat OpenShift Service on AWS (ROSA) cluster deployments, including information about cluster limits and scalability.
# Table of Contents

## CHAPTER 1. AWS PREREQUISITES FOR ROSA WITH STS  

1.1. DEPLOYMENT PREREQUISITES ....................................................... 3  
1.2. CUSTOMER REQUIREMENTS WHEN USING STS FOR DEPLOYMENT ........... 3  
   1.2.1. Account ........................................................................ 3  
   1.2.2. Access requirements ....................................................... 4  
   1.2.3. Support requirements ..................................................... 4  
   1.2.4. Security requirements .................................................... 4  
1.3. REQUIREMENTS FOR DEPLOYING A CLUSTER IN AN OPT-IN REGION ...... 4  
   1.3.1. Setting the AWS security token version ................................ 5  
1.4. RED HAT MANAGED IAM REFERENCES FOR AWS ......................... 5  
1.5. PROVISIONED AWS INFRASTRUCTURE ....................................... 6  
   1.5.1. EC2 instances ................................................................ 6  
   1.5.2. AWS Elastic Block Store (EBS) storage ............................... 6  
   1.5.3. Elastic load balancers ..................................................... 6  
   1.5.4. S3 storage .................................................................... 7  
   1.5.5. VPC  
   
   1.5.5.1. Sample VPC Architecture .......................................... 7  
   1.5.6. Security groups .............................................................. 7  
1.6. NEXT STEPS ............................................................................. 8  
1.7. ADDITIONAL RESOURCES ....................................................... 8  

## CHAPTER 2. LIMITS AND SCALABILITY  

2.1. INITIAL PLANNING CONSIDERATIONS ......................................... 9  
2.2. ROSA TESTED CLUSTER MAXIMUMS ......................................... 9  
2.3. OPENSSHIFT CONTAINER PLATFORM TESTING ENVIRONMENT AND CONFIGURATION ................................................................. 10  

## CHAPTER 3. PLANNING YOUR ENVIRONMENT  

3.1. PLANNING YOUR ENVIRONMENT BASED ON TESTED CLUSTER MAXIMUMS 12  
3.2. PLANNING YOUR ENVIRONMENT BASED ON APPLICATION REQUIREMENTS 12  

## CHAPTER 4. REQUIRED AWS SERVICE QUOTAS  

4.1. REQUIRED AWS SERVICE QUOTAS .............................................. 16  
4.2. NEXT STEPS ............................................................................. 17  

## CHAPTER 5. SETTING UP THE ENVIRONMENT FOR USING STS  

5.1. SETTING UP THE ENVIRONMENT FOR STS .................................... 18  
5.2. NEXT STEPS ............................................................................. 21  
5.3. ADDITIONAL RESOURCES ....................................................... 21
CHAPTER 1. AWS PREREQUISITES FOR ROSA WITH STS

Red Hat OpenShift Service on AWS (ROSA) provides a model that allows Red Hat to deploy clusters into a customer’s existing Amazon Web Service (AWS) account.

Ensure that the following AWS prerequisites are met before installing ROSA with STS.

1.1. DEPLOYMENT PREREQUISITES

To deploy Red Hat OpenShift Service on AWS (ROSA) into your existing Amazon Web Services (AWS) account, Red Hat requires that several prerequisites are met.

Red Hat recommends the use of AWS Organizations to manage multiple AWS accounts. The AWS Organizations, managed by the customer, host multiple AWS accounts. There is a root account in the organization that all accounts will refer to in the account hierarchy.

It is a best practice for the ROSA cluster to be hosted in an AWS account within an AWS Organizational Unit. A service control policy (SCP) is created and applied to the AWS Organizational Unit that manages what services the AWS sub-accounts are permitted to access. The SCP applies only to available permissions within a single AWS account for all AWS sub-accounts within the Organizational Unit. It is also possible to apply a SCP to a single AWS account. All other accounts in the customer’s AWS Organizations are managed in whatever manner the customer requires. Red Hat Site Reliability Engineers (SRE) will not have any control over SCPs within AWS Organizations.

**IMPORTANT**

When you create a ROSA cluster using AWS STS, an associated AWS OpenID Connect (OIDC) identity provider is created as well. This OIDC provider configuration relies on a public key that is located in the us-east-1 AWS region. Customers with AWS SCPs must allow the use of the us-east-1 AWS region, even if these clusters are deployed in a different region.

1.2. CUSTOMER REQUIREMENTS WHEN USING STS FOR DEPLOYMENT

The following prerequisites must be complete before you deploy a Red Hat OpenShift Service on AWS (ROSA) cluster that uses the AWS Security Token Service (STS).

1.2.1. Account

- The customer ensures that the AWS limits are sufficient to support Red Hat OpenShift Service on AWS provisioned within the customer’s AWS account.
- If SCP policies are applied and enforced, these policies must not be more restrictive than the roles and policies required by the cluster.
- The customer’s AWS account should not be transferable to Red Hat.
- The customer should not impose additional AWS usage restrictions beyond the defined roles and policies on Red Hat activities. Imposing restrictions will severely hinder Red Hat’s ability to respond to incidents.
- The customer may deploy native AWS services within the same AWS account.
The account must have a service-linked role set up as it is required for elastic load balancers (ELBs) to be configured.

**NOTE**

Customers are encouraged, but not mandated, to deploy resources in a Virtual Private Cloud (VPC) separate from the VPC hosting Red Hat OpenShift Service on AWS and other Red Hat supported services.

### 1.2.2. Access requirements

- Red Hat must have AWS console access to the customer-provided AWS account. This access is protected and managed by Red Hat.

- The customer must not utilize the AWS account to elevate their permissions within the Red Hat OpenShift Service on AWS cluster.

- Actions available in the *rosa* CLI utility or *OpenShift Cluster Manager* console must not be directly performed in the customer’s AWS account.

### 1.2.3. Support requirements

- Red Hat recommends that the customer have at least *Business Support* from AWS.

- Red Hat may have permission from the customer to request AWS support on their behalf.

- Red Hat may have permission from the customer to request AWS resource limit increases on the customer’s account.

- Red Hat manages the restrictions, limitations, expectations, and defaults for all Red Hat OpenShift Service on AWS clusters in the same manner, unless otherwise specified in this requirements section.

### 1.2.4. Security requirements

- Volume snapshots will remain within the customer’s AWS account and customer-specified region.

- Red Hat must have ingress access to EC2 hosts and the API server from allow-listed IP addresses.

- Red Hat must have egress allowed to the documented domains.

### 1.3. REQUIREMENTS FOR DEPLOYING A CLUSTER IN AN OPT-IN REGION

An AWS opt-in region is a region that is not enabled by default. If you want to deploy a Red Hat OpenShift Service on AWS (ROSA) cluster that uses the AWS Security Token Service (STS) in an opt-in region, you must meet the following requirements:

- The region must be enabled in your AWS account. For more information about enabling opt-in regions, see *Managing AWS Regions* in the AWS documentation.
The security token version in your AWS account must be set to version 2. You cannot use version 1 security tokens for opt-in regions.

**IMPORTANT**
Updating to security token version 2 can impact the systems that store the tokens, due to the increased token length. For more information, see the AWS documentation on setting STS preferences.

### 1.3.1. Setting the AWS security token version

If you want to create a Red Hat OpenShift Service on AWS (ROSA) cluster with the AWS Security Token Service (STS) in an AWS opt-in region, you must set the security token version to version 2 in your AWS account.

**Prerequisites**

- You have installed and configured the latest AWS CLI on your installation host.

**Procedure**

1. List the ID of the AWS account that is defined in your AWS CLI configuration:

   ```bash
   $ aws sts get-caller-identity --query Account --output json
   ``
   Ensure that the output matches the ID of the relevant AWS account.

2. List the security token version that is set in your AWS account:

   ```bash
   $ aws iam get-account-summary --query SummaryMap.GlobalEndpointTokenVersion --output json
   
   Example output
   
   1
   ``

3. To update the security token version to version 2 for all regions in your AWS account, run the following command:

   ```bash
   $ aws iam set-security-token-service-preferences --global-endpoint-token-version v2Token
   
   **IMPORTANT**
   Updating to security token version 2 can impact the systems that store the tokens, due to the increased token length. For more information, see the AWS documentation on setting STS preferences.

### 1.4. RED HAT MANAGED IAM REFERENCES FOR AWS

With the STS deployment model, Red Hat is no longer responsible for creating and managing Amazon Web Services (AWS) IAM policies, IAM users, or IAM roles.
1.5. PROVISIONED AWS INFRASTRUCTURE

This is an overview of the provisioned Amazon Web Services (AWS) components on a deployed Red Hat OpenShift Service on AWS (ROSA) cluster. For a more detailed listing of all provisioned AWS components, see the OpenShift Container Platform documentation.

1.5.1. EC2 instances

AWS EC2 instances are required for deploying the control plane and data plane functions of ROSA in the AWS public cloud.

Instance types can vary for control plane and infrastructure nodes, depending on the worker node count. At a minimum, the following EC2 instances will be deployed:

- Three m5.2xlarge control plane nodes
- Two r5.xlarge infrastructure nodes
- Two m5.xlarge customizable worker nodes

For further guidance on worker node counts, see the link to "Initial Planning Considerations" in the "Additional resources" section of this page.

1.5.2. AWS Elastic Block Store (EBS) storage

Amazon EBS block storage is used for both local node storage and persistent volume storage.

Volume requirements for each EC2 instance:

- Control Plane Volume
  - Size: 350GB
  - Type: io1
  - Input/Output Operations Per Second: 1000
- Infrastructure Volume
  - Size: 300GB
  - Type: gp2
  - Input/Output Operations Per Second: 900
- Worker Volume
  - Size: 300GB
  - Type: gp2
  - Input/Output Operations Per Second: 900

1.5.3. Elastic load balancers

Up to two Network Elastic Load Balancers (ELBs) for API and up to two Classic ELBs for application router. For more information, see the ELB documentation for AWS.
1.5.4. S3 storage

The image registry and Elastic Block Store (EBS) volume snapshots are backed by AWS S3 storage. Pruning of resources is performed regularly to optimize S3 usage and cluster performance.

**NOTE**

Two buckets are required with a typical size of 2TB each.

1.5.5. VPC

Customers should expect to see one VPC per cluster. Additionally, the VPC will need the following configurations:

- **Subnets**: Two subnets for a cluster with a single availability zone, or six subnets for a cluster with multiple availability zones.
- **Router tables**: One router table per private subnet, and one additional table per cluster.
- **Internet gateways**: One Internet Gateway per cluster.
- **NAT gateways**: One NAT Gateway per public subnet.

1.5.5.1. Sample VPC Architecture

1.5.6. Security groups

AWS security groups provide security at the protocol and port access level; they are associated with EC2 instances and Elastic Load Balancers. Each security group contains a set of rules that filter traffic coming in and out of an EC2 instance. You must ensure the ports required for the OpenShift installation are open on your network and configured to allow access between hosts.
<table>
<thead>
<tr>
<th>Group</th>
<th>Type</th>
<th>IP Protocol</th>
<th>Port range</th>
</tr>
</thead>
<tbody>
<tr>
<td>MasterSecurityGroup</td>
<td>AWS::EC2::Security Group</td>
<td>icmp</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tcp</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tcp</td>
<td>6443</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tcp</td>
<td>22623</td>
</tr>
<tr>
<td>WorkerSecurityGroup</td>
<td>AWS::EC2::Security Group</td>
<td>icmp</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tcp</td>
<td>22</td>
</tr>
<tr>
<td>BootstrapSecurityGroup</td>
<td>AWS::EC2::Security Group</td>
<td>tcp</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tcp</td>
<td>19531</td>
</tr>
</tbody>
</table>

1.6. NEXT STEPS

Review the required AWS service quotas

1.7. ADDITIONAL RESOURCES

- See Initial Planning Considerations for guidance on worker node count.
- See SRE access to all Red Hat OpenShift Service on AWS clusters for information about how Red Hat site reliability engineering accesses ROSA clusters.
CHAPTER 2. LIMITS AND SCALABILITY

2.1. INITIAL PLANNING CONSIDERATIONS

Consider the following tested object maximums when you plan your Red Hat OpenShift Service on AWS cluster.

These guidelines are based on a cluster of 102 workers in a multi-availability zone configuration. For smaller clusters, the maximums are lower.

The sizing of the control plane and infrastructure nodes is dynamically calculated during the installation process, based on the number of worker nodes. If you change the number of worker nodes after the installation, control plane and infra nodes must be resized manually. Infra nodes are resized by the Red Hat SRE team, and you can open a ticket in the Customer Portal to request the infra node resizing.

The following table lists the size of control plane and infrastructure nodes that are assigned during installation.

<table>
<thead>
<tr>
<th>Number of worker nodes</th>
<th>Control plane size</th>
<th>Infrastructure node size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 25</td>
<td>m5.2xlarge</td>
<td>r5.xlarge</td>
</tr>
<tr>
<td>26 to 100</td>
<td>m5.4xlarge</td>
<td>r5.2xlarge</td>
</tr>
<tr>
<td>101 to 180 (^1)</td>
<td>m5.8xlarge</td>
<td>r5.4xlarge</td>
</tr>
</tbody>
</table>

1. The maximum number of worker nodes on ROSA is 180

For larger clusters, infrastructure node sizing can become a large impacting factor to scalability. There are many factors that influence the stated thresholds, including the etcd version or storage data format.

Exceeding these limits does not necessarily mean that the cluster will fail. In most cases, exceeding these numbers results in lower overall performance.

The OpenShift Container Platform version used in all of the tests is OCP 4.8.0.

2.2. ROSA TESTED CLUSTER MAXIMUMS

The following table specifies the maximum limits for each tested type in a Red Hat OpenShift Service on AWS cluster.

Table 2.1. Tested cluster maximums

<table>
<thead>
<tr>
<th>Maximum type</th>
<th>4.8 tested maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>102</td>
</tr>
<tr>
<td>Number of pods (^1)</td>
<td>20,400</td>
</tr>
<tr>
<td>Number of pods per node</td>
<td>250</td>
</tr>
<tr>
<td>Maximum type</td>
<td>4.8 tested maximum</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Number of pods per core</td>
<td>There is no default value</td>
</tr>
<tr>
<td>Number of namespaces</td>
<td>3,400</td>
</tr>
<tr>
<td>Number of pods per namespace</td>
<td>20,400</td>
</tr>
<tr>
<td>Number of services</td>
<td>10,000</td>
</tr>
<tr>
<td>Number of services per namespace</td>
<td>10,000</td>
</tr>
<tr>
<td>Number of back ends per service</td>
<td>10,000</td>
</tr>
<tr>
<td>Number of deployments per namespace</td>
<td>1,000</td>
</tr>
</tbody>
</table>

1. The pod count displayed here is the number of test pods. The actual number of pods depends on the application’s memory, CPU, and storage requirements.

2. When there are a large number of active projects, etcd can suffer from poor performance if the keyspace grows excessively large and exceeds the space quota. Periodic maintenance of etcd, including defragmentation, is highly recommended to make etcd storage available.

3. There are a number of control loops in the system that must iterate over all objects in a given namespace as a reaction to some changes in state. Having a large number of objects of a type, in a single namespace, can make those loops expensive and slow down processing the state changes. The limit assumes that the system has enough CPU, memory, and disk to satisfy the application requirements.

4. Each service port and each service back end has a corresponding entry in iptables. The number of back ends of a given service impacts the size of the endpoints objects, which then impacts the size of data that is sent throughout the system.

In OpenShift Container Platform 4.8, half of a CPU core (500 millicore) is reserved by the system compared to previous versions of OpenShift Container Platform.

2.3. OPENSHIFT CONTAINER PLATFORM TESTING ENVIRONMENT AND CONFIGURATION

The following table lists the OpenShift Container Platform environment and configuration on which the cluster maximums are tested for the AWS cloud platform.

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>vCPU</th>
<th>RAM(GiB)</th>
<th>Disk type</th>
<th>Disk size(GiB) /I0S</th>
<th>Count</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control plane/etc</td>
<td>m5.4xlarge</td>
<td>16</td>
<td>64</td>
<td>io1</td>
<td>350 / 1,000</td>
<td>3</td>
<td>us-west-2</td>
</tr>
</tbody>
</table>
CHAPTER 2. LIMITS AND SCALABILITY

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>vCPU</th>
<th>RAM(GiB)</th>
<th>Disk type</th>
<th>Disk size(GiB) / IOPS</th>
<th>Count</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure nodes</td>
<td>r5.2xlarge</td>
<td>8</td>
<td>64</td>
<td>gp2</td>
<td>300 / 900</td>
<td>3</td>
<td>us-west-2</td>
</tr>
<tr>
<td>Workload nodes</td>
<td>m5.2xlarge</td>
<td>8</td>
<td>32</td>
<td>gp2</td>
<td>350 / 900</td>
<td>3</td>
<td>us-west-2</td>
</tr>
<tr>
<td>Worker nodes</td>
<td>m5.2xlarge</td>
<td>8</td>
<td>32</td>
<td>gp2</td>
<td>350 / 900</td>
<td>102</td>
<td>us-west-2</td>
</tr>
</tbody>
</table>

1. io1 disks are used for control plane/etcd nodes because etcd is I/O intensive and latency sensitive. A greater number of IOPS can be required, depending on usage.

2. Infrastructure nodes are used to host monitoring components because Prometheus can claim a large amount of memory, depending on usage patterns.

3. Workload nodes are dedicated to run performance and scalability workload generators.

Larger cluster sizes and higher object counts might be reachable. However, the sizing of the infrastructure nodes limits the amount of memory that is available to Prometheus. When creating, modifying, or deleting objects, Prometheus stores the metrics in its memory for roughly 3 hours prior to persisting the metrics on disk. If the rate of creation, modification, or deletion of objects is too high, Prometheus can become overwhelmed and fail due to the lack of memory resources.
CHAPTER 3. PLANNING YOUR ENVIRONMENT

3.1. PLANNING YOUR ENVIRONMENT BASED ON TESTED CLUSTER MAXIMUMS

This document describes how to plan your Red Hat OpenShift Service on AWS environment based on the tested cluster maximums.

Oversubscribing the physical resources on a node affects resource guarantees the Kubernetes scheduler makes during pod placement. Learn what measures you can take to avoid memory swapping.

Some of the tested maximums are stretched only in a single dimension. They will vary when many objects are running on the cluster.

The numbers noted in this documentation are based on Red Hat testing methodology, setup, configuration, and tunings. These numbers can vary based on your own individual setup and environments.

While planning your environment, determine how many pods are expected to fit per node using the following formula:

\[
\frac{\text{required pods per cluster}}{\text{pods per node}} = \text{total number of nodes needed}
\]

The current maximum number of pods per node is 250. However, the number of pods that fit on a node is dependent on the application itself. Consider the application’s memory, CPU, and storage requirements, as described in Planning your environment based on application requirements.

Example scenario

If you want to scope your cluster for 2200 pods per cluster, you would need at least nine nodes, assuming that there are 250 maximum pods per node:

\[
\frac{2200}{250} = 8.8
\]

If you increase the number of nodes to 20, then the pod distribution changes to 110 pods per node:

\[
\frac{2200}{20} = 110
\]

Where:

\[
\frac{\text{required pods per cluster}}{\text{total number of nodes}} = \text{expected pods per node}
\]

3.2. PLANNING YOUR ENVIRONMENT BASED ON APPLICATION REQUIREMENTS

This document describes how to plan your Red Hat OpenShift Service on AWS environment based on your application requirements.

Consider an example application environment:
<table>
<thead>
<tr>
<th>Pod type</th>
<th>Pod quantity</th>
<th>Max memory</th>
<th>CPU cores</th>
<th>Persistent storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>apache</td>
<td>100</td>
<td>500 MB</td>
<td>0.5</td>
<td>1 GB</td>
</tr>
<tr>
<td>node.js</td>
<td>200</td>
<td>1 GB</td>
<td>1</td>
<td>1 GB</td>
</tr>
<tr>
<td>postgresql</td>
<td>100</td>
<td>1 GB</td>
<td>2</td>
<td>10 GB</td>
</tr>
<tr>
<td>JBoss EAP</td>
<td>100</td>
<td>1 GB</td>
<td>1</td>
<td>1 GB</td>
</tr>
</tbody>
</table>

Extrapolated requirements: 550 CPU cores, 450 GB RAM, and 1.4 TB storage.

Instance size for nodes can be modulated up or down, depending on your preference. Nodes are often resource overcommitted. In this deployment scenario, you can choose to run additional smaller nodes or fewer larger nodes to provide the same amount of resources. Factors such as operational agility and cost-per-instance should be considered.

<table>
<thead>
<tr>
<th>Node type</th>
<th>Quantity</th>
<th>CPUs</th>
<th>RAM (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes (option 1)</td>
<td>100</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>Nodes (option 2)</td>
<td>50</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>Nodes (option 3)</td>
<td>25</td>
<td>16</td>
<td>64</td>
</tr>
</tbody>
</table>

Some applications lend themselves well to overcommitted environments, and some do not. Most Java applications and applications that use huge pages are examples of applications that would not allow for overcommitment. That memory can not be used for other applications. In the example above, the environment would be roughly 30 percent overcommitted, a common ratio.

The application pods can access a service either by using environment variables or DNS. If using environment variables, for each active service the variables are injected by the kubelet when a pod is run on a node. A cluster-aware DNS server watches the Kubernetes API for new services and creates a set of DNS records for each one. If DNS is enabled throughout your cluster, then all pods should automatically be able to resolve services by their DNS name. Service discovery using DNS can be used in case you must go beyond 5000 services. When using environment variables for service discovery, if the argument list exceeds the allowed length after 5000 services in a namespace, then the pods and deployments will start failing.

Disable the service links in the deployment’s service specification file to overcome this:

**Example**

```
Kind: Template
apiVersion: template.openshift.io/v1
metadata:
  name: deploymentConfigTemplate
creationTimestamp: 
annotations:
```
This template will create a deploymentConfig with 1 replica, 4 env vars and a service.

```yaml
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
metadata:
  name: deploymentconfig${IDENTIFIER}
spec:
  template:
    metadata:
      labels:
        name: replicationcontroller${IDENTIFIER}
    spec:
      enableServiceLinks: false
      containers:
      - name: pause${IDENTIFIER}
        image: "${IMAGE}"
        ports:
          - containerPort: 8080
            protocol: TCP
        env:
        - name: ENVVAR1_${IDENTIFIER}
          value: "${ENV_VALUE}"
        - name: ENVVAR2_${IDENTIFIER}
          value: "${ENV_VALUE}"
        - name: ENVVAR3_${IDENTIFIER}
          value: "${ENV_VALUE}"
        - name: ENVVAR4_${IDENTIFIER}
          value: "${ENV_VALUE}"
resources: {}
        imagePullPolicy: IfNotPresent
        capabilities: {}
        securityContext:
          capabilities: {}
          privileged: false
        restartPolicy: Always
        serviceAccount: ""
      replicas: 1
      selector:
        name: replicationcontroller${IDENTIFIER}
    triggers:
    - type: ConfigChange
    strategy:
      type: Rolling
    - kind: Service
      apiVersion: v1
      metadata:
        name: service${IDENTIFIER}
      spec:
        selector:
          name: replicationcontroller${IDENTIFIER}
        ports:
        - name: serviceport${IDENTIFIER}
          protocol: TCP
          port: 80
          targetPort: 8080
```

Red Hat OpenShift Service on AWS 4 Prepare your environment
The number of application pods that can run in a namespace is dependent on the number of services and the length of the service name when the environment variables are used for service discovery. **ARG_MAX** on the system defines the maximum argument length for a new process and it is set to 2097152 KiB by default. The kubelet injects environment variables into each pod scheduled to run in the namespace including:

- `<SERVICE_NAME>_SERVICE_HOST=<IP>`
- `<SERVICE_NAME>_SERVICE_PORT=<PORT>`
- `<SERVICE_NAME>_PORT=tcp://<IP>:<PORT>`
- `<SERVICE_NAME>_PORT_<PORT>_TCP=tcp://<IP>:<PORT>`
- `<SERVICE_NAME>_PORT_<PORT>_TCP_PROTO=tcp`
- `<SERVICE_NAME>_PORT_<PORT>_TCP_PORT=<PORT>`
- `<SERVICE_NAME>_PORT_<PORT>_TCP_ADDR=<ADDR>`

The pods in the namespace start to fail if the argument length exceeds the allowed value and if the number of characters in a service name impacts it.
CHAPTER 4. REQUIRED AWS SERVICE QUOTAS

Review this list of the required Amazon Web Service (AWS) service quotas that are required to run a Red Hat OpenShift Service on AWS cluster.

4.1. REQUIRED AWS SERVICE QUOTAS

The table below describes the AWS service quotas and levels required to create and run a Red Hat OpenShift Service on AWS cluster.

NOTE

The AWS SDK allows ROSA to check quotas, but the AWS SDK calculation does not include your existing usage. Therefore, it is possible that the quota check can pass in the AWS SDK yet the cluster creation can fail. To fix this issue, increase your quota.

If you need to modify or increase a specific quota, see Amazon’s documentation on requesting a quota increase.

<table>
<thead>
<tr>
<th>Quota name</th>
<th>Service code</th>
<th>Quota code</th>
<th>Minimum required value</th>
<th>Recommended value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of EIPs - VPC EIPs</td>
<td>ec2</td>
<td>L-0263D0A3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Running On-Demand Standard (A, C, D, H, I, M, R, T, Z) instances</td>
<td>ec2</td>
<td>L-1216C47A</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>VPCs per Region</td>
<td>vpc</td>
<td>L-F678F1CE</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Internet gateways per Region</td>
<td>vpc</td>
<td>L-A4707A72</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Network interfaces per Region</td>
<td>vpc</td>
<td>L-DF5E4CA3</td>
<td>5,000</td>
<td>5,000</td>
</tr>
<tr>
<td>General Purpose SSD (gp2) volume storage</td>
<td>ebs</td>
<td>L-D18FCD1D</td>
<td>50</td>
<td>300</td>
</tr>
<tr>
<td>Number of EBS snapshots</td>
<td>ebs</td>
<td>L-309BACF6</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>Provisioned IOPS</td>
<td>ebs</td>
<td>L-B3A130E6</td>
<td>300,000</td>
<td>300,000</td>
</tr>
<tr>
<td>Provisioned IOPS SSD (io1) volume storage</td>
<td>ebs</td>
<td>L-FD252861</td>
<td>50</td>
<td>300</td>
</tr>
<tr>
<td>Quota name</td>
<td>Service code</td>
<td>Quota code</td>
<td>Minimum required value</td>
<td>Recommended value</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>----------------</td>
<td>-------------</td>
<td>------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Application Load Balancers per Region</td>
<td>elasticloadbalancing</td>
<td>L-53DA6B97</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Classic Load Balancers per Region</td>
<td>elasticloadbalancing</td>
<td>L-E9E9831D</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

4.2. NEXT STEPS

- Set up the environment and install ROSA
CHAPTER 5. SETTING UP THE ENVIRONMENT FOR USING STS

After you meet the AWS prerequisites, set up your environment and install Red Hat OpenShift Service on AWS (ROSA).

5.1. SETTING UP THE ENVIRONMENT FOR STS

Complete the following steps to set up your environment before creating your cluster using AWS security token Service (STS).

Prerequisites

- Review and complete the deployment prerequisites and policies.
- Create a Red Hat account, if you do not already have one. Then, check your email for a verification link. You will need these credentials to install ROSA.

Procedure

1. Log in to the Amazon Web Services (AWS) account that you want to use.
   It is recommended to use a dedicated AWS account to run production clusters. If you are using AWS Organizations, you can use an AWS account within your organization or create a new one.

   If you are using AWS Organizations and you need to have a service control policy (SCP) applied to the AWS account you plan to use, these policies must not be more restrictive than the roles and policies required by the cluster.

2. Enable the ROSA service in the AWS Console.
   a. Sign in to your AWS account.
   b. To enable ROSA, go to the ROSA service and select Enable OpenShift.

3. Install and configure the AWS CLI.
   a. Follow the AWS command-line interface documentation to install and configure the AWS CLI for your operating system.
      Specify the correct `aws_access_key_id` and `aws_secret_access_key` in the `.aws/credentials` file. See AWS Configuration basics in the AWS documentation.
   b. Set a default AWS region.

      **NOTE**
      You can use the environment variable to set the default AWS region.

      The ROSA service evaluates regions in the following priority order:

      i. The region specified when running a `rosa` command with the `--region` flag.
      ii. The region set in the `AWS_DEFAULT_REGION` environment variable. See Environment variables to configure the AWS CLI in the AWS documentation.
iii. The default region set in your AWS configuration file. See Quick configuration with `aws configure` in the AWS documentation.

c. Optional: Configure your AWS CLI settings and credentials by using an AWS named profile. `rosa` evaluates AWS named profiles in the following priority order:

   i. The profile specified when running a `rosa` command with the `--profile` flag.

   ii. The profile set in the `AWS_PROFILE` environment variable. See Named profiles in the AWS documentation.

d. Verify the AWS CLI is installed and configured correctly by running the following command to query the AWS API:

   ```
   $ aws sts get-caller-identity
   ```

4. Install the latest version of `rosa`, the Red Hat OpenShift Service on AWS command-line interface (CLI).

   a. Download the latest release of the `rosa` CLI for your operating system.

   b. Optional: Rename the file you downloaded to `rosa` and make the file executable. This documentation uses `rosa` to refer to the executable file.

      ```
      $ chmod +x rosa
      ```

   c. Optional: Add `rosa` to your path.

      ```
      $ mv rosa /usr/local/bin/rosa
      ```

   d. Enter the following command to verify your installation:

      ```
      $ rosa
      ```

Example output

Command line tool for ROSA.

Usage:

  rosa [command]

Available Commands:

  completion  Generates bash completion scripts
  create      Create a resource from stdin
  delete      Delete a specific resource
  describe    Show details of a specific resource
  edit        Edit a specific resource
  help        Help about any command
  init        Applies templates to support Managed OpenShift on AWS clusters
  list        List all resources of a specific type
  login       Log in to your Red Hat account
  logout      Log out
  logs        Show logs of a specific resource
  verify      Verify resources are configured correctly for cluster install
  version     Prints the version of the tool
Generate the command completion scripts for the `rosa` CLI. The following example generates the Bash completion scripts for a Linux machine:

```bash
$ rosa completion bash | sudo tee /etc/bash_completion.d/rosa
```

Source the scripts to enable `rosa` command completion from your existing terminal. The following example sources the Bash completion scripts for `rosa` on a Linux machine:

```bash
$ source /etc/bash_completion.d/rosa
```

5. Log in to your Red Hat account with the `rosa` CLI.
   
a. Enter the following command.

```bash
$ rosa login
```

b. Replace `<my_offline_access_token>` with your token.

**Example output**

To login to your Red Hat account, get an offline access token at https://console.redhat.com/openshift/token/rosa

? Copy the token and paste it here: `<my-offline-access-token>`

**Example output continued**

I: Logged in as '<rh-rosa-user>' on 'https://api.openshift.com'

6. Verify that your AWS account has the necessary quota to deploy an Red Hat OpenShift Service on AWS cluster.

```bash
$ rosa verify quota [--region=<region>]
```

**Example output**

I: Validating AWS quota...
I: AWS quota ok

**NOTE**

Sometimes your AWS quota varies by region. If you receive any errors, try a different region.
If you need to increase your quota, go to your AWS console, and request a quota increase for the service that failed.

After both the permissions and quota checks pass, proceed to the next step.

7. Prepare your AWS account for cluster deployment:
   a. Run the following command to verify your Red Hat and AWS credentials are setup correctly. Check that your AWS Account ID, Default Region and ARN match what you expect. You can safely ignore the rows beginning with OpenShift Cluster Manager for now.

   ```
   $ rosa whoami
   ```

   **Example output**

   ```
   AWS Account ID:               000000000000
   AWS Default Region:           us-east-1
   AWS ARN:                      arn:aws:iam::000000000000:user/hello
   OCM API:                      https://api.openshift.com
   OCM Account ID:               1DzGldhqEWyl8UUQXhSoWaaaaa
   OCM Account Name:             Your Name
   OCM Account Username:         you@domain.com
   OCM Account Email:            you@domain.com
   OCM Organization ID:          1HopHfA2hcmhup5gCr2uH5aaaaa
   OCM Organization Name:        Red Hat
   OCM Organization External ID: 0000000
   ```

8. Install the OpenShift CLI (`oc`), version 4.7.9 or greater, from the ROSA (`rosa`) CLI.
   a. Enter this command to download the latest version of the `oc` CLI:

   ```
   $ rosa download openshift-client
   ```

   b. After downloading the `oc` CLI, unzip it and add it to your path.

   c. Enter this command to verify that the `oc` CLI is installed correctly:

   ```
   $ rosa verify openshift-client
   ```

Create roles

After completing these steps, you are ready to set up IAM and OIDC access-based roles.

5.2. NEXT STEPS

- Create a ROSA cluster with STS quickly or create a cluster using customizations.

5.3. ADDITIONAL RESOURCES

- AWS Prerequisites
- Required AWS service quotas and increase requests