APCERA

Apcera Training

APC101 - Introducing the Apcera Platform

Lab Book

Apcera Training

Labs Overview

This document lists the lab exercises that comprise Apcera Training.

Lab Setup

First, you are going to perform a few tasks to set up your environment for the labs. These tasks include downloading the sample apps, installing the APC tool, targeting your cluster, exploring the Apcera Web Console, accessing your cluster documentation, and familiarizing yourself with additional learning resources.

APC Set Up for Mac OS X, Linux, and Windows Clients

Complete the following steps.

Task 1: Clone (copy) the Apcera sample apps repository using command line

If you are familiar with the git commands, follow the instructions here. Otherwise, please **skip** to Task 2.

Step	Instruction
A.1.1	Launch a command line session and issue the following command: `git clone https://github.com/apcera/sample-apps` The repository will be copied to whichever directory the git command is ran from, for example: **`/Users/<username>/sample-apps`**
A.1.2	Go to Task 3

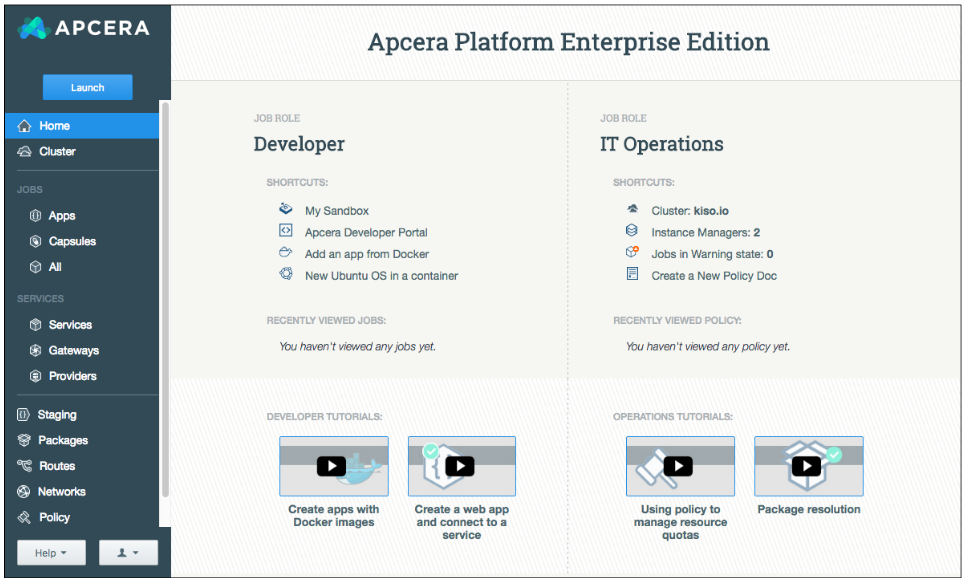
Task 2: Clone (copy) the Apcera sample apps repository using a browser

Note: If you have already completed Task 1, please skip to Task 3.

Step	Instruction
A.2.1	Open a web browser, navigate to the following URL: https://github.com/apcera/sample-apps
A.2.2	Click the **Download ZIP** button and save the *sample-apps-master.zip* archive to your computer.

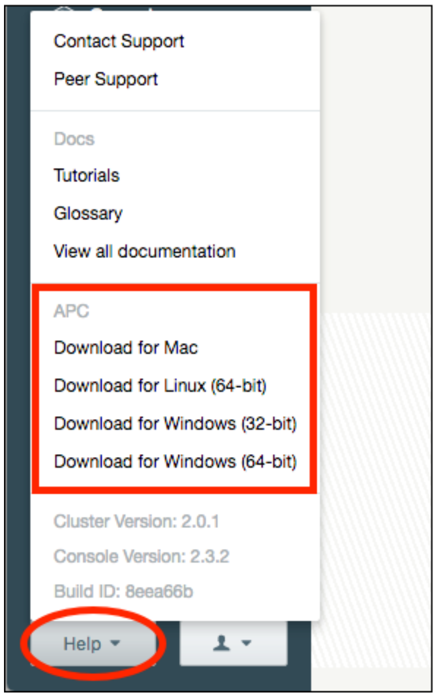
A.2.3	Extract the ZIP file to a directory such as: **/Users/<username>/sample-apps**.

Task 3: Log in to the Web Console using your Google credentials

Step	Instruction
A.3.1	The Web Console lets you manage and monitor the apps deployed to your cluster. Use your browser to access the console at the following URL: https://console.kiso.io.
A.3.2	Log in using your Google credentials. Once logged in, take a minute to explore the console. You may want to bookmark the console URL for convenience.  Apcera Platform Enterprise Edition JOB ROLE Developer SHORTCUTS: My Sandbox Apcera Developer Portal Add an app from Docker New Ubuntu OS in a container RECENTLY VIEWED JOBS: *You haven't viewed any jobs yet.* JOB ROLE IT Operations SHORTCUTS: Cluster: **kiso.io** Instance Managers: **2** Jobs in Warning state: **0** Create a New Policy Doc RECENTLY VIEWED POLICY: *You haven't viewed any policy yet.* DEVELOPER TUTORIALS: Create apps with Docker images Create a web app and connect to a service OPERATIONS TUTORIALS: Using policy to manage resource quotas Package resolution **NOTE:** If you cannot log in to the console, clear your browser's cache and try again.

Task 4: Install APC command line tool

APC is the command line interface (CLI) tool that you use to interact with your Apcera cluster.

Step	Instruction
A.4.1	In the Web Console, click the **Help** button to expand its menu, and then select the applicable APC download for your OS.  Contact Support Peer Support Docs Tutorials Glossary View all documentation APC Download for Mac Download for Linux (64-bit) Download for Windows (32-bit) Download for Windows (64-bit) Cluster Version: 2.0.1 Console Version: 2.3.2 Build ID: 8eea66b Help
A.4.2	**Mac OS X users ONLY:** Double-click the PKG file and follow the installer instructions.
A.4.3	**Linux users ONLY:** Unzip the downloaded file to find the apc.exe. NOTE: If necessary you can set optional environment variables for APC. Refer to the following documentation: http://docs.apcera.com/quickstart/installing-apc/.
A.4.4	**Windows users ONLY:** Double click the apc.exe to run your APC command line tool. NOTE: For your convenience, add apc.exe to your PATH environment variable.

Task 5: Login to the lab cluster

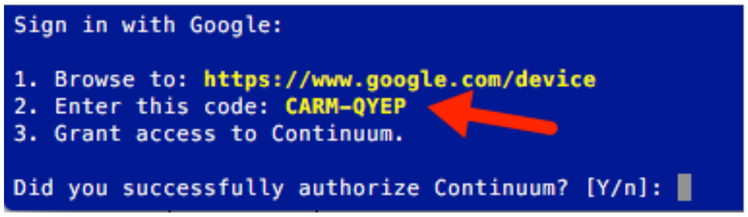
Important: Your cluster is configured by policy to authenticate you using Google Device auth. The instructor will add your **Google Gmail address** to the cluster authentication policy. Please provide the instructor with your Google Gmail address so it can be added to the policy. If you do not have a Gmail address, please create one now and give it to the instructor.

Once you have given the instructor your Gmail address, use a browser and log in to your Gmail account.

Step	Instruction
A.5.1	Open a command window, execute the following `apc` command to verify that the APC is correctly installed. `apc version` If APC is installed, this command will return the version of APC that you are running. If no version is returned, reinstall APC.

A.5.2	For this training a cluster has been set up for you. The training cluster name is ***kiso.io***. To access your cluster using APC, you target it. Once targeted, that cluster is the one your APC session interfaces with until you change the target.
	
	Execute the following command to target your cluster:
	
	`apc target kiso.io`
	
	Success message:
	`Targeted [https://kiso.io]`
	`Logged in as: "<username>"`
	
	Notice your user name.

A.5.3	Execute the following command to log in: `apc login`
A.5.4	Browse to https://www.google.com/device

A.5.5	Select your Google account, and then enter the unique 8-letter code provided by APC.
	
	`Sign in with Google:`
	`1. Browse to: https://www.google.com/device`
	`2. Enter this code: CARM-QYEP`
	`3. Grant access to Continuum.`
	`Did you successfully authorize Continuum? [Y/n]:`

A.5.6	Click **Allow** to grant access to the cluster.
A.5.7	Return to the command window, and type `y` (yes) at the prompt, “Did you successfully authorized Continuum?” and press enter. Success message: ***`Login successful for <username>`***
A.5.8	To ensure that you are logged in as your Google user, execute the following command: `apc login show` The “Current login” should be your Google Gmail user name. If it is not, log out of the cluster and redo the above steps. Do not proceed until you are logged in as your Google user.

A.5.9	Execute the following command to set the local namespace to be your home namespace as defined by the authentication policy:
	
	`apc namespace --default`
	
	Expected result: You should see the following messages:
	`Setting namespace to default '/sandbox/<username>'... done`
	`Success!`
	
	NOTE: If your default namespace is different than *'/sandbox/<username>'*, please notify the instructor and get his or her help before proceeding.

Task 6: Update your APC version (if necessary)

Step	Instruction
A.6.1	Issue the following command to verify that your APC version is in sync with the cluster: `apc update` You should receive the message “No new updates available.” NOTE: If your APC client version is out of sync with the cluster, Apcera will notify you of this. Follow the instructions shown by APC to update to the correct version.

Task 7: Familiarize yourself with the APC documentation

Step	Instruction
A.7.1	Once you have verified your APC installation, issue the following command: `apc help` You should see the message “APC is Continuum's command-line tool,” followed by usage syntax, subcommands, and optional arguments.

A.7.2

Use the following syntax to access the APC documentation for a particular command:

```
apc help <subcommand1> <subcommand2>
```

For example:

```
apc help app
apc help app create
apc help job
apc help job list
```

Notice that APC provides two ways to specify most optional arguments: long form and short form. For example, you can use either of the following to list all jobs in the root namespace:

Long form: **`apc job list --namespace /`**
Short form: **`apc job list -ns /`**

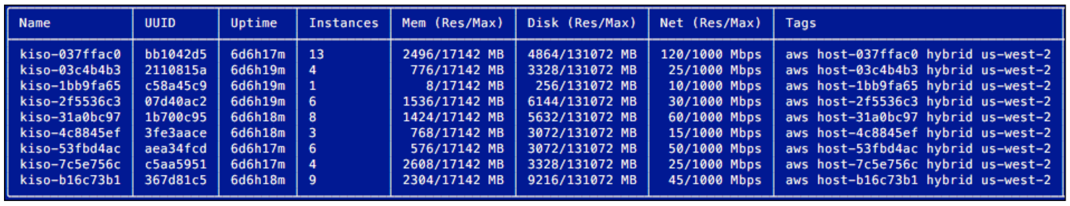
The long form uses a double dash. The short form uses a single dash and an abbreviation. A few options do not have shortcuts (for example, --start). This training uses the long form.

A.7.3

Run the following command to view the list of instance managers that are able to run jobs:

```
apc cluster instance-manager list
```

The **kiso.io** cluster manages nine Amazon EC2 instances running in US West 2 region of AWS.



```
Name           UUID      Uptime   Instances  Mem (Res/Max)   Disk (Res/Max)   Net (Res/Max)  Tags
kiso-037ffac0  bb1042d5  6d6h17m  13         2496/17142 MB   4864/131072 MB   120/1000 Mbps  aws host-037ffac0 hybrid us-west-2
kiso-03c4b4b3  2110815a  6d6h19m  4           776/17142 MB   3328/131072 MB    25/1000 Mbps  aws host-03c4b4b3 hybrid us-west-2
kiso-1bb9fa65  c58a45c9  6d6h19m  1             8/17142 MB    256/131072 MB    10/1000 Mbps  aws host-1bb9fa65 hybrid us-west-2
kiso-2f5536c3  07d40ac2  6d6h19m  6          1536/17142 MB   6144/131072 MB    30/1000 Mbps  aws host-2f5536c3 hybrid us-west-2
kiso-31a0bc97  1b700c95  6d6h18m  8          1424/17142 MB   5632/131072 MB    60/1000 Mbps  aws host-31a0bc97 hybrid us-west-2
kiso-4c8845ef  3fe3aace  6d6h18m  3           768/17142 MB   3072/131072 MB    15/1000 Mbps  aws host-4c8845ef hybrid us-west-2
kiso-53fbd4ac  aea34fcd  6d6h17m  6           576/17142 MB   3072/131072 MB    50/1000 Mbps  aws host-53fbd4ac hybrid us-west-2
kiso-7c5e756c  c5aa5951  6d6h17m  4          2608/17142 MB   3328/131072 MB    25/1000 Mbps  aws host-7c5e756c hybrid us-west-2
kiso-b16c73b1  367d81c5  6d6h18m  9          2304/17142 MB   9216/131072 MB    45/1000 Mbps  aws host-b16c73b1 hybrid us-west-2
```

The Apcera documentation is publicly available at http://docs.apcera.com/. In addition, each cluster hosts a version of the Apcera documentation. You can access your cluster documentation at http://docs.kiso.io/, or by by clicking the question mark icon (?) at the upper right of the console and selecting *View all documentation*.

With these preliminary tasks complete, you are now ready to proceed to Lab 1, where you will learn how to use Apcera to deploy a diverse set of workloads, including apps, Docker images, and OS capsules.

APCERA

Apcera Training

Lab 1: Govern with Policy

Lab Introduction

In this lab you author policies that permit you to use the system. First you write a policy that allows you to create jobs and packages in your namespace. Next you update the policy to allow you to create services, and then define a rule pair that lets you bind a job to a service. Lastly, you write a rule that defines the base operating system (OS) for capsules.

Lab Exercise

This lab comprises the following exercises:

A. Define policy to authorize app creation (job and package)
B. Define policy to authorize service creation and bindings

Lab Prerequisites

This lab assumes that you have set up your lab environment. If you have not done so, please follow the lab setup guide now.

Related Documentation

- Using Policy to Govern Cluster Resources: http://docs.apcera.com/policy/policy-toc/

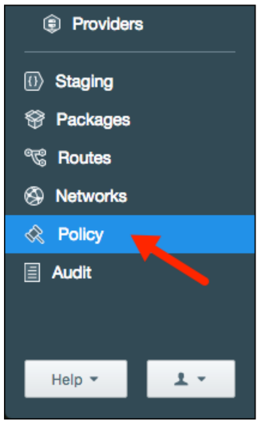
Lab Instruction

For each exercise, complete the steps.

Exercise A: Define policy and rules for app creation

In this exercise, you define a policy and policy rules that allow you to create applications in your sandbox. To create apps, you need permissions for both the job and package realms.

Task 1: Author policy that permits you to deploy apps

Step	Instruction
A.1.1	Log in to the Web Console using your Google credentials.
A.1.2	Select the **Policy** from the left menu.  Providers Staging Packages Routes Networks Policy Audit Help You should see a policy document named **user-policy-<*user_name*>**, where <*user_name*> is <u>your user name</u>. This is a policy document that you can use to author policy for your namespace /sandbox/<user_name>.
A.1.3	Select the **user-policy-<user_name>** document.
A.1.4	Click the **Edit Policy** button.

A.1.5	Enter the following policy rule beneath the commented section.
	`on job::/sandbox/<user_name> {`
	`    if (auth_server@apcera.me->name == <user_name>) {`
	`        permit create, read, update, delete`
	`        permit start, stop, ssh, link, bind, map`
	`    }`
	`}`

	NOTE: Substitute “**<user_name>**” with your Apcera username (here and elsewhere)
A.1.6	Click **Apply Changes** when you are done. The Policy Editor validates the syntax for you.

Task 2: Deploy a Java application

Step	Instruction

A.2.1	Using APC, target your cluster and login:
	
	`apc target kiso.io`
	
	`apc login`

A.2.2	Change your working directory to /sample-apps/example-java-hello. `cd /sample-apps/example-java-hello`
A.2.3	Deploy the program using the following command: `apc app create hello-world -dr --batch` Expected results: App creation **fails**. Error: Not allowed to create "package::/sandbox/<*username*>::hello-world" by policy: missing claim "permit create" on "package::/sandbox/<*username*>::hello-world" Why? Because you do not have the necessary permissions. To deploy apps, as a developer you need to both job and package resources and thus need permissions on both realms.

A.2.4

Update your policy to include package resource permissions:

```
on job::/sandbox/<user_name> {
      if (auth_server@apcera.me->name == <user_name>) {
            permit create, read, update, delete
            permit start, stop, ssh, link, map
      }
}

on package::/sandbox/<user_name> {
    if (auth_server@apcera.me->name == <user_name>) {
        role admin
    }
}
```

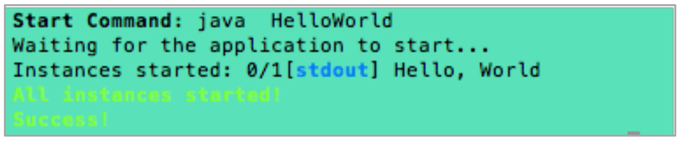
The first rule permits CRUD operations on jobs in your namespace. The second rule assigns you to the admin role on the package realm. **Why the difference?**

A.2.5

Try to create the Java app again:

```
apc app create hello-world -dr --batch --start
```

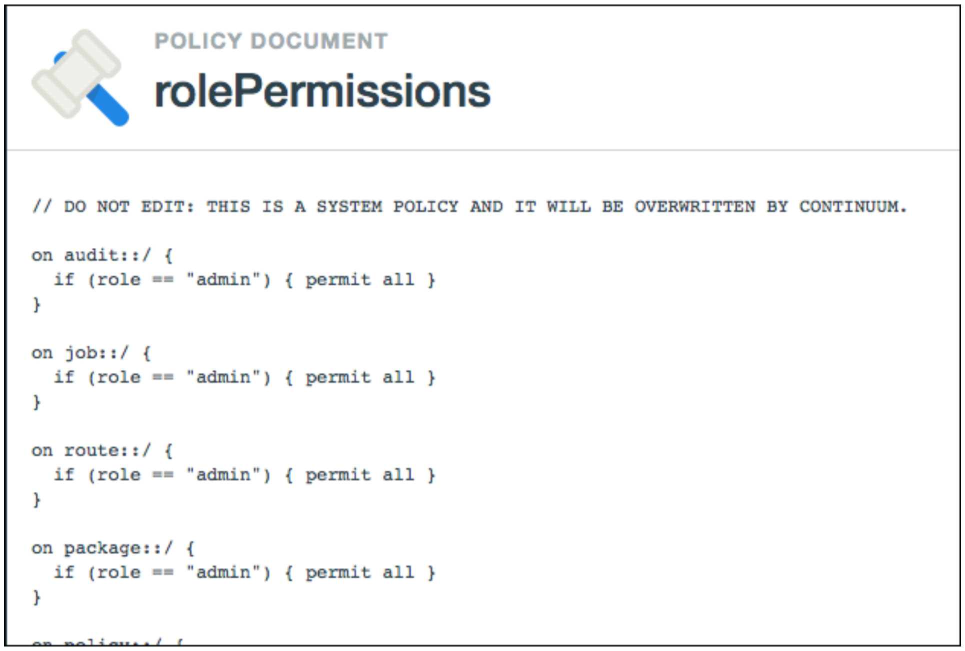
This time the app should be successfully created because you have written and applied the necessary policy permissions for creating jobs and packages in your namespace.



```
Start Command: java  HelloWorld
Waiting for the application to start...
Instances started: 0/1[stdout] Hello, World
All instances started!
Success!
```

This demonstrates how Apcera implements attribute-based access control.

In the **system policy**, it is defined what the role ‘**admin**’ authorizes for job and package realms:

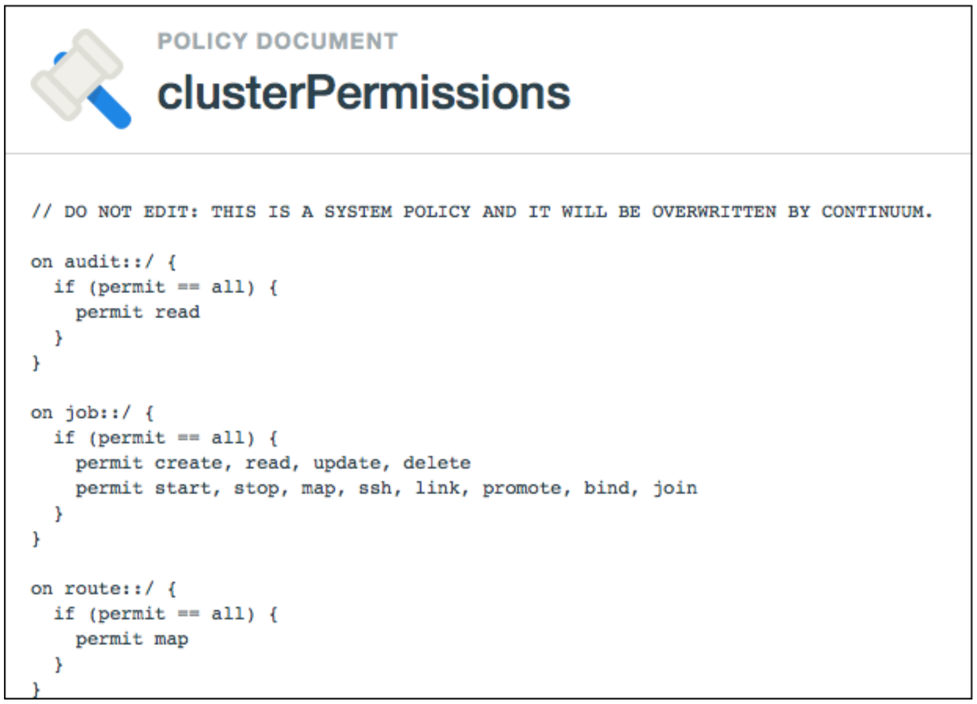


POLICY DOCUMENT

rolePermissions

```
// DO NOT EDIT: THIS IS A SYSTEM POLICY AND IT WILL BE OVERWRITTEN BY CONTINUUM.

on audit::/ {
  if (role == "admin") { permit all }
}

on job::/ {
  if (role == "admin") { permit all }
}

on route::/ {
  if (role == "admin") { permit all }
}

on package::/ {
  if (role == "admin") { permit all }
}
```

And what ‘**permit all**’ means for a job:

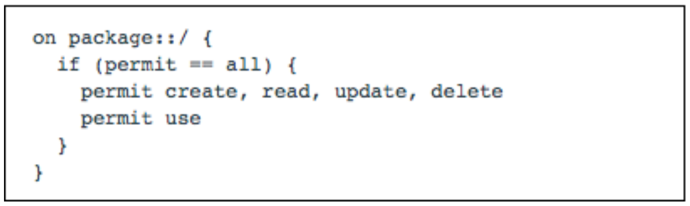


POLICY DOCUMENT

clusterPermissions

```
// DO NOT EDIT: THIS IS A SYSTEM POLICY AND IT WILL BE OVERWRITTEN BY CONTINUUM.

on audit::/ {
  if (permit == all) {
    permit read
  }
}

on job::/ {
  if (permit == all) {
    permit create, read, update, delete
    permit start, stop, map, ssh, link, promote, bind, join
  }
}

on route::/ {
  if (permit == all) {
    permit map
  }
}
```

And for a package:



```
on package::/ {
  if (permit == all) {
    permit create, read, update, delete
    permit use
  }
}
```

Thus for packages in your namespace, you have all permissions via the admin role. But for jobs, you do not have all permissions. This shows the attribute-based access control capabilities of the system, the granularity with which you can exert control over workloads, and then extensibility of the policy language.

Task 3: Review user-<user_name> Policy

The **user-<*user_name*>** policy is already in place so you can do the labs. But it is important to understand how permissions are tightly controlled within the system using realms.

The following rule permits the issuance of a token in your name after you are authenticated.

```
auth::/oauth2/http {
  if (Google->email == "continuum.<user_name>.2015@gmail.com") {
     permit issue
     name "<user_name>"
  }
}
```

The following rule assigns you a default namesapce:

```
auth::/ {
  if (auth_server@apcera.me->name == "<user_name>") {
     defaultNamespace "/sandbox/<user_name>"
  }
}
```

The following rule gives you a policy document in that namespace:

```
policydoc::/::user-policy-<user_name> {
  if (auth_server@apcera.me->name == "<user_name>") {
     permit read, update
  }
}
```

Exercise B: Define policy rules for service creation

In this exercise, you add rules to create services and bindings in your namespace. Service creation will fail without policy permissions to create services.

Task 1: Author a policy to allow service creation

Step	Instruction
B.1.1	Select **user-policy-<*user_name*>** policy, and click the **Edit Policy** button.

B.1.2	Add the following rule to the bottom of your policy.
	
	`on service::/sandbox/<user_name> {`
	`    if (auth_server@apcera.me->name=="<user_name>"){`
	`        role admin`
	`    }`
	`}`

B.1.3	Click **Apply Changes** to save.

NOTE: The role permissions are:

```
on service::/ {
  if (role == "admin") { permit all }
}
```

And the cluster permissions are:

```
on service::/ {
  if (permit == all) {
      permit create, read, update, delete
      permit bind
  }
}
```

Therefore, the policy you created in Step B.1.2 was equivalent to permitting create, read, update, delete, and bind on the service::/sandbox/<user_name> realm.

Task 2: Explore and update the job policy

In a subsequent lab you will create a service and bind a job to it. If you were to do so with the policies you have in place, this process would not be permitted by policy. ***Why?***

For services, you need permissions to create and bind that service. The policy rule on the service realm authorizes service creation and binding, so ***what is the problem?***

```
Error: Not allowed to bind "job::/sandbox/<user_name>::myapp" to
"service::/sandbox/<user_name>::mydb" by policy:

missing claim "permit bind" on "job::/sandbox/<user_name>::myapp"
```

A binding is a credentialed connection between a job and a service. Binding is a two-way policy handshake. Both the service and job must permit it. At this point the policy rule you created for the job realm does not permit binding.

Step	Instruction

B.2.1	Update the job rule to permit job binding, linking, and mapping, permissions that are associated with connecting your app to other jobs, services, and routes.
	
	You can do this by adding each permission explicitly, or by giving yourself admin permissions on the job realm.
	
	`on job::/sandbox/<user_name> {`
	`      if (auth_server@apcera.me->name == "<user_name>") {`
	`            permit create, read, update, delete`
	`            permit start, stop, ssh, link, map`
	`            permit bind`
	`      }`
	`}`
	
	Or…
	
	`on job::/sandbox/<user_name>  {`
	`     if (auth_server@apcera.me->name == "<user_name>") {`
	`          role admin`
	`     }`
	`}`
	
	What is the difference between the rules? What does one allow but not the other?

In **production**, you would likely want to have different permission levels for finer granularity, such as the following.

```
on service::/sandbox/<user_name> {
    if (auth_server@apcera.me->name=="<user_name>"){
        permit create, read
    }
}

on service::/sandbox/<user_name> {
    if (job nameMatch "job::/sandbox/<user_name>") {
        permit bind
    }
}

on job::/sandbox/<user_name> {
    if (service nameMatch "service::/sandbox/<user_name>") {
        permit bind
    }
}
```

If desired, comment out the above policies you created for jobs and services and create finer-grained rules for these resources. As you progress through the labs, you may run across policy permission issues requiring you to update your policy rules. Try doing so, and if you get stuck enable (uncomment) the admin or fuller-permission rules you have defined.

Exercise Review

In this lab you wrote a policy and rules that allowed you create jobs, packages, and services, and bind jobs and services. In addition, you learned how to use policy for package resolution.

If you completed this lab successfully, your policy should have the following rules:

```
on job::/sandbox/<user_name> {
      if (auth_server@apcera.me->name == <user_name>) {
            permit create, read, update, delete
            permit start, stop, ssh, link, map
            permit bind
      }
}

on package::/sandbox/<user_name> {
      if (auth_server@apcera.me->name == <user_name>) {
            role admin
      }
}

on service::/sandbox/<user_name> {
      if (auth_server@apcera.me->name == <user_name>) {
            role admin
      }
}
```

End of the Lab

Apcera Training

Lab 2: Advanced Policy

Lab Introduction

In this lab you author policies that permit you to use the system. First you write a policy that allows you to create jobs and packages in your namespace. Next you update the policy to allow you to create services, and then define a rule pair that lets you bind a job to a service. Lastly, you write a rule that defines the base operating system (OS) for capsules.

Lab Exercise

This lab comprises the following exercises:

A. Define policy for package resolution
B. Define policy to whitelist Docker repositories

Lab Prerequisites

This lab assumes that you have set up your lab environment. If you have not done so, please follow the lab setup guide now.

Related Documentation

- Using Policy to Govern Cluster Resources: http://docs.apcera.com/policy/policy-toc/

Lab Instruction

For each exercise, complete the steps.

Apcera Training

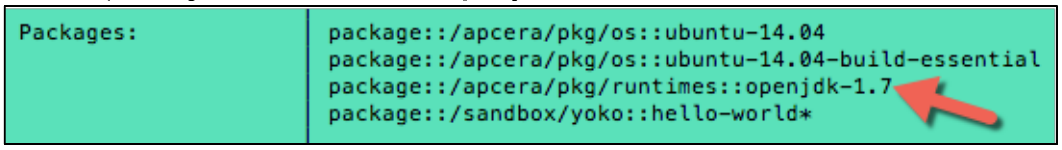
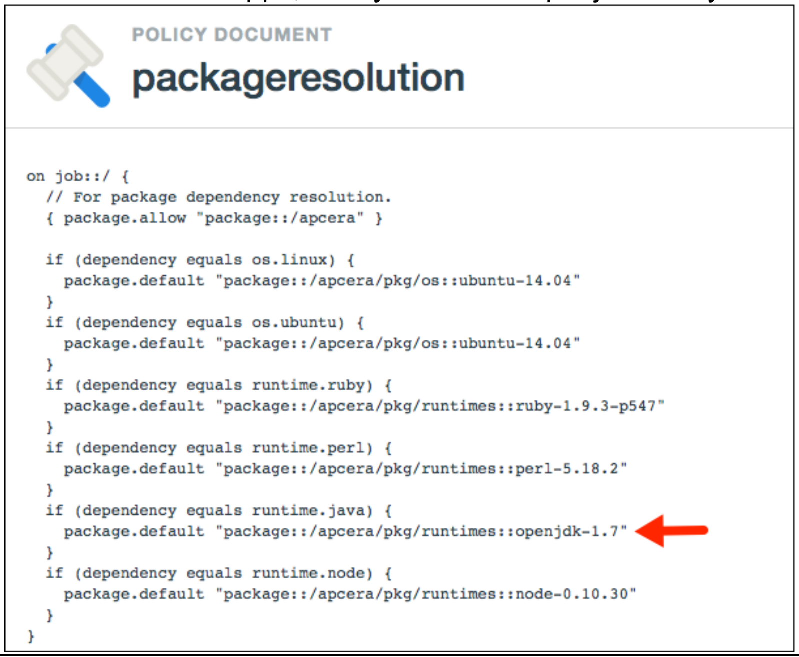
Lab 2: Advanced Policy

Exercise A: Create Policy for Package Resolution

A capsule is a bare OS container. In this exercise, you use policy to set the base OS for capsule creation in your sandbox to be different from the default.

Task 1: Overwrite the default package rule

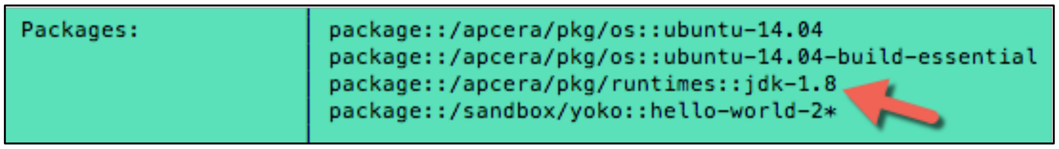
Step	Instruction

A.1.1	Execute the following command to list the details about hello-world app you created earlier.
	
	`apc app show hello-world`
	
	This command lists the packages that this app depends on. Notice that the runtime package that was used is **openjdk-1.7**.
	
	`Packages:  package::/apcera/pkg/os::ubuntu-14.04`
	`package::/apcera/pkg/os::ubuntu-14.04-build-essential`
	`package::/apcera/pkg/runtimes::openjdk-1.7`
	`package::/sandbox/yoko::hello-world*`
	
	NOTE: The **packageResolution** policy sets the system package default for the clutser. For Java apps, the system uses openjdk-1.7 by default.
	
	POLICY DOCUMENT
	packageresolution
	
	`on job::/ {`
	`  // For package dependency resolution.`
	`  { package.allow "package::/apcera" }`
	
	`  if (dependency equals os.linux) {`
	`    package.default "package::/apcera/pkg/os::ubuntu-14.04"`
	`  }`
	`  if (dependency equals os.ubuntu) {`
	`    package.default "package::/apcera/pkg/os::ubuntu-14.04"`
	`  }`
	`  if (dependency equals runtime.ruby) {`
	`    package.default "package::/apcera/pkg/runtimes::ruby-1.9.3-p547"`
	`  }`
	`  if (dependency equals runtime.perl) {`
	`    package.default "package::/apcera/pkg/runtimes::perl-5.18.2"`
	`  }`
	`  if (dependency equals runtime.java) {`
	`    package.default "package::/apcera/pkg/runtimes::openjdk-1.7"`
	`  }`
	`  if (dependency equals runtime.node) {`
	`    package.default "package::/apcera/pkg/runtimes::node-0.10.30"`
	`  }`
	`}`

A.1.2	Return to the console, select **user-policy-<user_name>** policy, and then click the **Edit Policy** button.

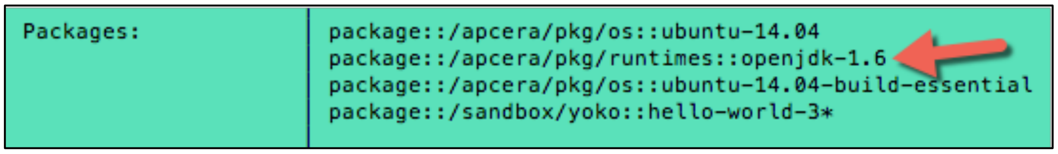
A.1.3	Add the following rule in your policy to set the Java runtime default to JDK 1.8.
	`on job::/sandbox/<user_name> {`
	`  if (dependency equals runtime.java)`
	`  {`
	`     package.default "package::/apcera/pkg/runtimes::jdk-1.8"`
	`  }`
	`}`
	NOTE: This sets the Java runtime default to jdk-1.8 in your namespace, `/sandbox/<user_name>`. The system default is still openjdk-1.7 for other namespaces.

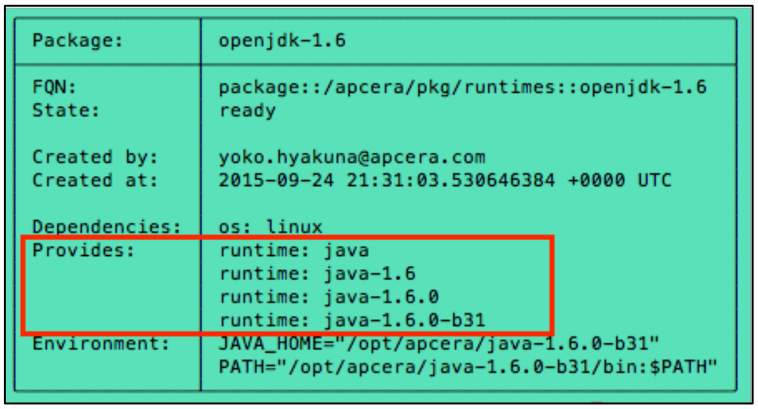
A.1.4	Click **Apply Changes** to save.
A.1.5	Make sure that your working directory is still /continuum-sample-apps/example-java-hello. Create a second app. `apc app create hello-world-2 -dr --batch --start`

A.1.6	Execute the following command to list the details about hello-world-2 app.
	`apc app show hello-world-2`
	Verify that the hello-world-2 app uses **jdk-1.8** instead of openjdk-1.7.
	
	`Packages:  package::/apcera/pkg/os::ubuntu-14.04`
	`           package::/apcera/pkg/os::ubuntu-14.04-build-essential`
	`           package::/apcera/pkg/runtimes::jdk-1.8`
	`           package::/sandbox/yoko::hello-world-2*`

Task 2: Retiring a package

Step	Instruction
A.2.1	Execute the following command to create the Java hello app with JDK 1.6. `apc app create hello-world-3 --depends-on runtime.java-1.6 -dr --batch --start` NOTE: The **`--depends-on`** option specifies dependencies for the application with the form **<type>.<name>** where <type> is 'os', 'package', 'file', or 'runtime'. In this case, you are requiring runtime package for java-1.6 to create hello-world-3.

A.2.2	List the app detail: `apc app show hello-world-3`
	
	
	`Packages:  package::/apcera/pkg/os::ubuntu-14.04`
	`package::/apcera/pkg/runtimes::openjdk-1.6`
	`package::/apcera/pkg/os::ubuntu-14.04-build-essential`
	`package::/sandbox/yoko::hello-world-3*`

A.2.3	Execute the following command to view the openjdk-1.6 package details.
	
	`apc package show /apcera/pkg/runtimes::openjdk-1.6`
	
	
	`Package: openjdk-1.6`
	`FQN: package::/apcera/pkg/runtimes::openjdk-1.6`
	`State: ready`
	`Created by: yoko.hyakuna@apcera.com`
	`Created at: 2015-09-24 21:31:03.530646384 +0000 UTC`
	`Dependencies: os: linux`
	`Provides: runtime: java`
	`runtime: java-1.6`
	`runtime: java-1.6.0`
	`runtime: java-1.6.0-b31`
	`Environment: JAVA_HOME="/opt/apcera/java-1.6.0-b31"`
	`PATH="/opt/apcera/java-1.6.0-b31/bin:$PATH"`
	
	Notice that the openjdk-1.6 package provides runtime environment for java, java-1.6, java-1.6.0, and java-1.6.0-b31. Therefore, if you create an app with **`--depends-on runtime.java-1.6.0`** option will use this openjdk-1.6 package.

A.2.4	Return to the console, and then click the **Edit Policy** button.

A.2.5	Add the following rule in your policy to retire JDK 1.6.
	
	`job::/sandbox/<user_name> {`
	`  {`
	`     package.retire "package::/apcera/pkg/runtimes::openjdk-1.6"`
	`  }`
	`}`
	
	NOTE: Notice that this policy rule has no if-then construct. This implies that the package is retired regardless (the condition is always true).

A.2.6	Click **Apply Changes** to save.

A.2.7	Execute the following command to create the app with Java 1.6.
	
	`apc app create hello-world-4 --depends-on runtime.java-1.6 -dr --batch --start`
	
	Expected Result:
	You should see an error stating that there is no package to satisfy the dependency on Java 1.6.
	`Packaging... done`
	`Creating package "hello-world-4"... done`
	`Uploading package contents... done!`
	`[staging] Subscribing to the staging process...`
	`[staging] Beginning staging with 'stagpipe::/apcera::java' pipeline, 1 stager(s) defined.`
	`[staging] Error: no packages satisfy dependency "runtime.java-1.6"`
	`[staging] Error: Failed to clone stager job`
	`[staging] Staging pipeline has been aborted`
	`[staging] "package::/sandbox/yoko::hello-world-4" has failed to stage`
	`Package "hello-world-4" deleted. Error: Staging has failed.`
	``Try `apc help` for more information.``

Exercise B: Create policy to authorize to run Docker images

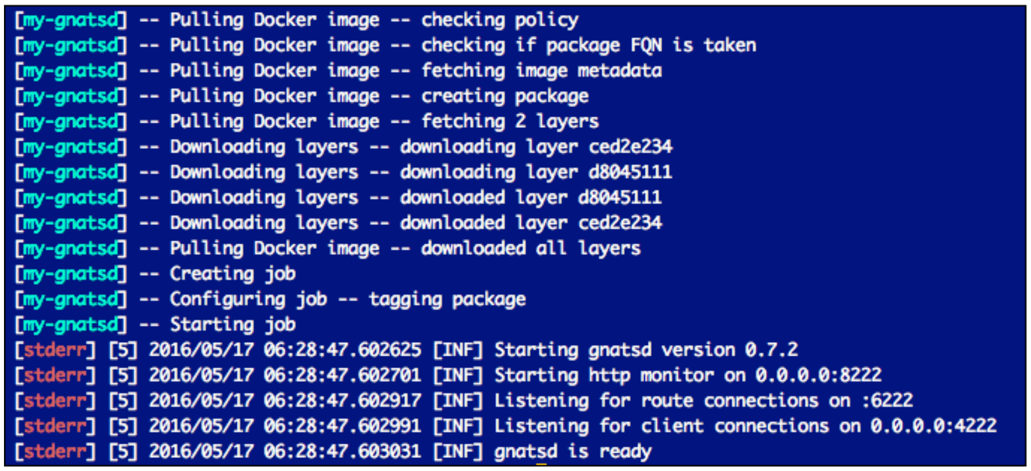
In this exercise, you define a policy and policy rules that allow you to run a Docker image in your sandbox.

Task 1: Create a policy to set

Step	Instruction
B.1.1	Execute the following command to run a Docker image. `apc docker run my-gnatsd --image apcera/gnatsd` Expected results: The command fails. **`Error: Not allowed to create job with origin "https://index.docker.io/apcera/gnatsd:latest": missing claim "docker.allow \"https://index.docker.io/apcera/gnatsd:latest\"" on "job::/sandbox/<user_name>::my-gnatsd"`** Why? Because you do not have the necessary permissions. To run a Docker image as a job, you need to provide additional permissions to whitelist the Docker Hub images that you wish to run.

B.1.2	Using the Policy Editor, add the following policy rule:
	
	`job::/sandbox/<user_name> {`
	`    if (auth_server@apcera.me->name == "<user_name>")`
	`    {`
	`     docker.allow "https://registry-1.docker.io/apcera"`
	`    }`
	`}`
	
	Using `docker.allow`, you whitelist a specific Docker Hub URL. The `docker.cache` allows the Docker image to be cached in your namespace.

B.1.3	Click **Apply Changes** to save.

B.1.4	Run the command again to verify that you can run a Docker image as a job.
	
	`apc docker run my-gnatsd --image apcera/gnatsd`
	
	You should see the following output:
	
	
	`[my-gnatsd] -- Pulling Docker image -- checking policy`
	`[my-gnatsd] -- Pulling Docker image -- checking if package FQN is taken`
	`[my-gnatsd] -- Pulling Docker image -- fetching image metadata`
	`[my-gnatsd] -- Pulling Docker image -- creating package`
	`[my-gnatsd] -- Pulling Docker image -- fetching 2 layers`
	`[my-gnatsd] -- Downloading layers -- downloading layer ced2e234`
	`[my-gnatsd] -- Downloading layers -- downloading layer d8045111`
	`[my-gnatsd] -- Downloading layers -- downloaded layer d8045111`
	`[my-gnatsd] -- Downloading layers -- downloaded layer ced2e234`
	`[my-gnatsd] -- Pulling Docker image -- downloaded all layers`
	`[my-gnatsd] -- Creating job`
	`[my-gnatsd] -- Configuring job -- tagging package`
	`[my-gnatsd] -- Starting job`
	`[stderr] [5] 2016/05/17 06:28:47.602625 [INF] Starting gnatsd version 0.7.2`
	`[stderr] [5] 2016/05/17 06:28:47.602701 [INF] Starting http monitor on 0.0.0.0:8222`
	`[stderr] [5] 2016/05/17 06:28:47.602917 [INF] Listening for route connections on :6222`
	`[stderr] [5] 2016/05/17 06:28:47.602991 [INF] Listening for client connections on 0.0.0.0:4222`
	`[stderr] [5] 2016/05/17 06:28:47.603031 [INF] gnatsd is ready`

B.1.5	Execute the following command to delete the Docker image you created. `apc job delete my-gnatsd`

B.1.6	Execute the following command to delete the apps you created.
	
	`apc app delete hello-world`
	`apc app delete hello-world-2`
	`apc app delete hello-world-3`

Exercise Review

In this lab you wrote a policy and rules that allowed you create jobs, packages, and services, and bind jobs and services. In addition, you learned how to use policy for package resolution.

If you completed this lab successfully, your policy should have the following rules:

```
on job::/sandbox/<user_name> {
    if(dependency equals runtime.java)
    {
        package.default "package::/apcera/pkg/runtimes::jdk-1.8"
    }
}

on job::/sandbox/<user_name> {
    {
        package.retire "package::/apcera/pkg/runtimes::openjdk-
1.6"
    }

}

on job::/sandbox/<user_name> {
    if (auth_server@apcera.me->name == <user_name>) {
        docker.allow "https://registry-1.docker.io/apcera"
     }
}
```

End of the Lab