

# SPINNAKER

DevelopIntelligence

## Our purpose...

We help organizations learn and adopt new technologies.



## ...Impacts you daily.

When you talk on the phone, watch a movie, connect with friends on social media, drive a car, fly on a plane, pay with a credit card, shop online, and order a latte with your mobile app, you are interacting with technology developed by one of our customers.

In 2018 alone...





Jenkins

akka



ANGULAR



APACHE  
Spark

aws

Azure

C#

cassandra

CHEF

D3

docker



=GO



GraphQL



JS

kafka



mongoDB

MySQL

node

puppet

python™



React Native



Scala

spring®



250 best selling books authored



9+ years of training experience



750,000 practitioner led training hours



EXPERT PRACTITIONERS

SEASONED CONSULTANTS

ENGAGING INSTRUCTOR

150 engagements speaking at industry conferences



Over 17 years of industry experience per instructor



125 certifications in leading technologies



95% instructor satisfaction

# VIRTUAL TRAINING



What we want



...what we've got

# VIRTUAL TRAINING EXPECTATIONS FOR ME

I pledge to:

- Make this as interesting and interactive as possible
- Ask questions in order to stimulate discussion
- Use whatever resources I have at hand to explain the material
- Try my best to manage verbal responses so that everyone who wants to speak can do so
- Use an on-screen timer for breaks so you know when to be back

# VIRTUAL TRAINING EXPECTATIONS FOR YOU

- Arrive and Return on Time
- Mute unless speaking
- Use chat or ask questions verbally

# AGENDA

## *Our mandatory goals for this class:*

- Become familiar with Spinnaker and it's role
- Understand the UI of Spinnaker
- Understand an Application, Pipeline, and Stage
- Purpose of Spinnaker
- How to get Spinnaker within Salesforce
- Helm and Baking
- Spinnaker Expression Language
- Understand the Role of Terraform

## SOME OTHER ITEMS YOU'LL PICK UP

- Canary
- Understanding of Kubernetes
- How it works
- Debugging Tips

# CONTINUOUS INTEGRATION AND DELIVERY

# WHY CONTINUOUS INTEGRATION?

- Continuous Feedback

# WHAT IS CONTINUOUS INTEGRATION?

- Taking code and integrating and testing with every check in!
- Employs Continuous Integration (CI) Server
  - Jenkins
  - Bamboo
  - Cruise Control
  - Travis

- Pre Deploy
  - Code Stability
  - Code Quality
  - Code Coverage
  - Security Analysis
- Post Deploy
  - Smoke
  - API

# WHY CONTINUOUS DELIVERY?

Giving power to business to release features as per wish

# WHAT IS CONTINUOUS DELIVERY?

Continuous delivery is based on the use of smart automation. this is all about creating a repeatable and reliable process for delivering software with an manual intervention.

# HOW WE IMPLEMENT CONTINUOUS DELIVERY?

- Delivering release to business
  - CI
  - QA validation
  - Perf validation
  - Security validation
  - UAT validation
- Manual trigger of shipping release to Production systems

## WHY CONTINUOUS DEPLOYMENT?

No Ops, Zero touch or GitOps shipping of release to production systems.

## WHAT IS CONTINUOUS DEPLOYMENT?

Continuous Deployment is a step up from Continuous Delivery in which every change in the source code is deployed to production automatically, without explicit approval from a developer.

# HOW DO WE IMPLEMENT CONTINUOUS DEPLOYMENT?

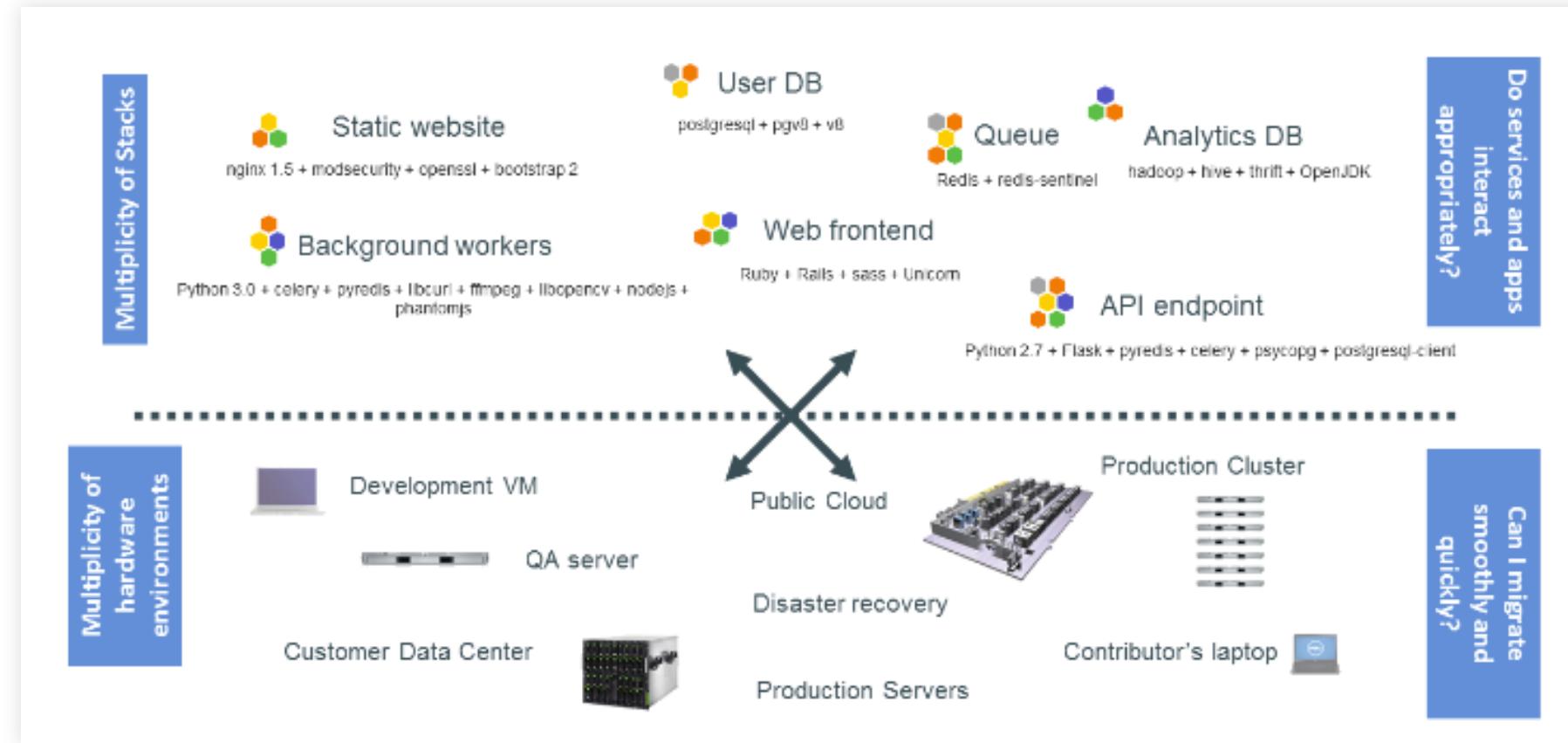
- Continuous Delivery
- Automated deployment to production systems as well

## EXTENDING CI WITH SPINNAKER

- Spinnaker takes content created in CI and deploys to staging and production
- Jenkins could do that to a point:
  - It does not manage multiple servers too well
  - Jenkins (not JenkinsX) doesn't work that well with Kubernetes

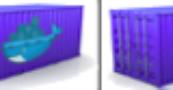
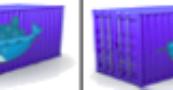
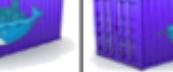
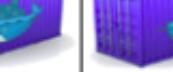
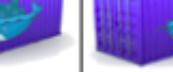
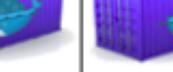
# KUBERNETES

# WHY MICROSERVICES?



	Static website	?	?	?	?	?	?
	Web frontend	?	?	?	?	?	?
	Background workers	?	?	?	?	?	?
	User DB	?	?	?	?	?	?
	Analytics DB	?	?	?	?	?	?
	Queue	?	?	?	?	?	?
	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers



	Static website						
	Web frontend						
	Background workers						
	User DB						
	Analytics DB						
	Queue						
	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers

# REVIEW OF KUBERNETES RESOURCES

# POD

- Group of one or more containers (such as Docker containers)
- Shared storage/network
- Specification for how to run the containers

**Question: How we will ensure HA & Scalability?**

# REPLICASETS

- A ReplicaSet purpose is to maintain a stable set of Pods count running at any given time.
- As such, it is often used to guarantee the availability of a specified number of identical Pods.

**Question: How we will release next version of our image?**

# DEPLOYMENT

- Provides declarative updates for Pods and ReplicaSets.
- Describe a desired state in a Deployment
- Deployment Controller changes the actual state to the desired state at a controlled rate.
- You can define Deployments to create new ReplicaSets,
- Or remove existing Deployments and adopt all their resources with new Deployment

**Question: How consumers of a microservice interact with pods?**

# SERVICE

- Service is an abstraction which defines a logical set of Pods and a policy by which to access them
- Sometimes this pattern is called a micro-service).
- The set of Pods targeted by a Service is usually determined by a selector

**Question: How we will expose our service to outside world?**

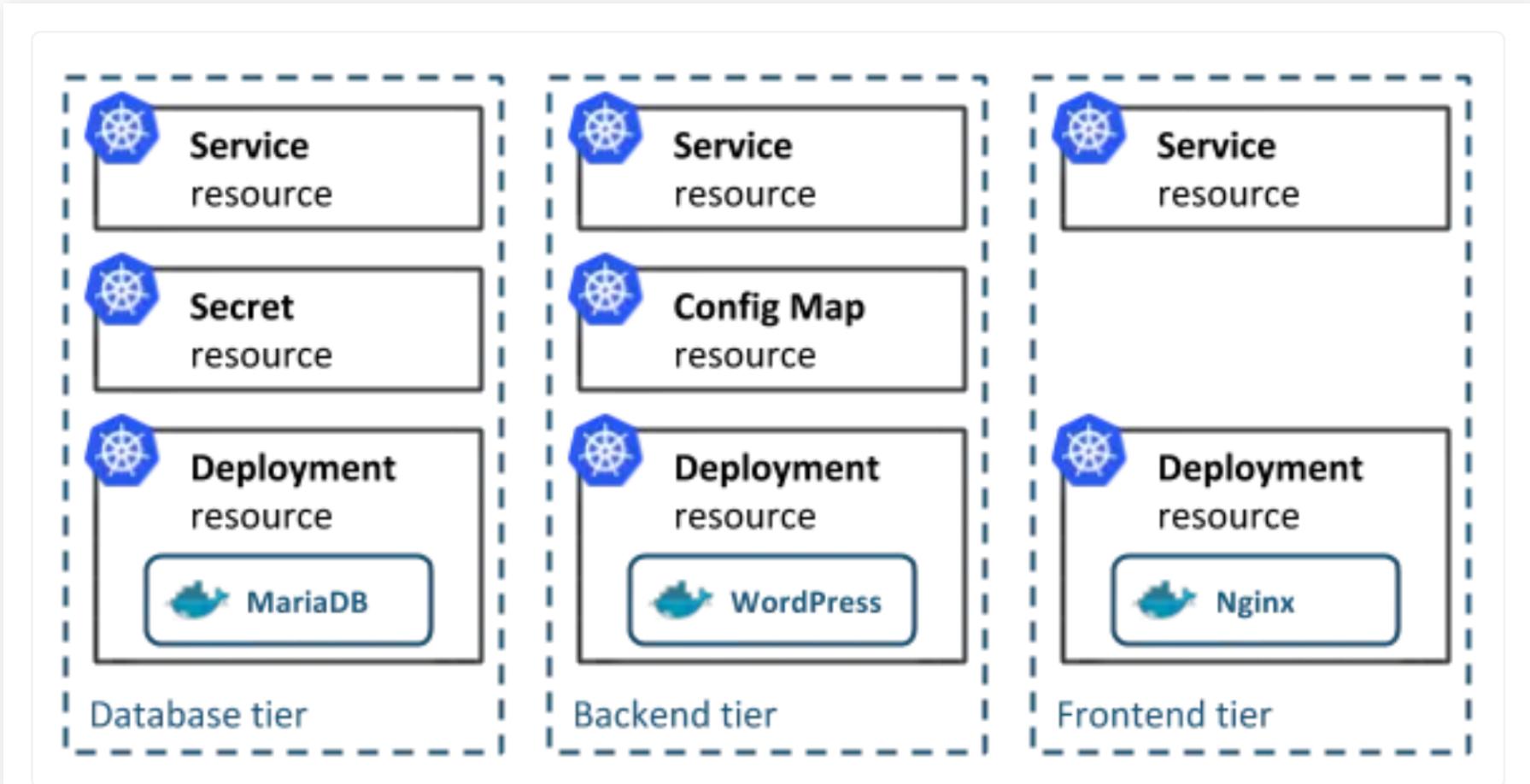
# INGRESS

- Object defines rules for routing HTTP(S) traffic to applications running in a cluster.
- An Ingress object is associated with one or more Service objects
- Each Service is associated with a set of Pods

**Question: How can we run the same application across different environments, without building image again and again?**

# CONFIGMAP

- Bind to your Pods' containers and system components at runtime
  - Configuration files
  - Command-line arguments
  - Environment variables
  - Port numbers
  - Other configuration artifacts
- Enable you to separate your configurations from your Pods and components, making workloads portable.



# OVERVIEW

# WHAT IS SPINNAKER?

- Open Source, Multicloud Platform
- Release Software at high velocity and confidence
- Extends where Jenkins CI leaves off
- Manages Staging and Deployment
- Makes use of containers
- Provides Baking Ability and Canary Ability

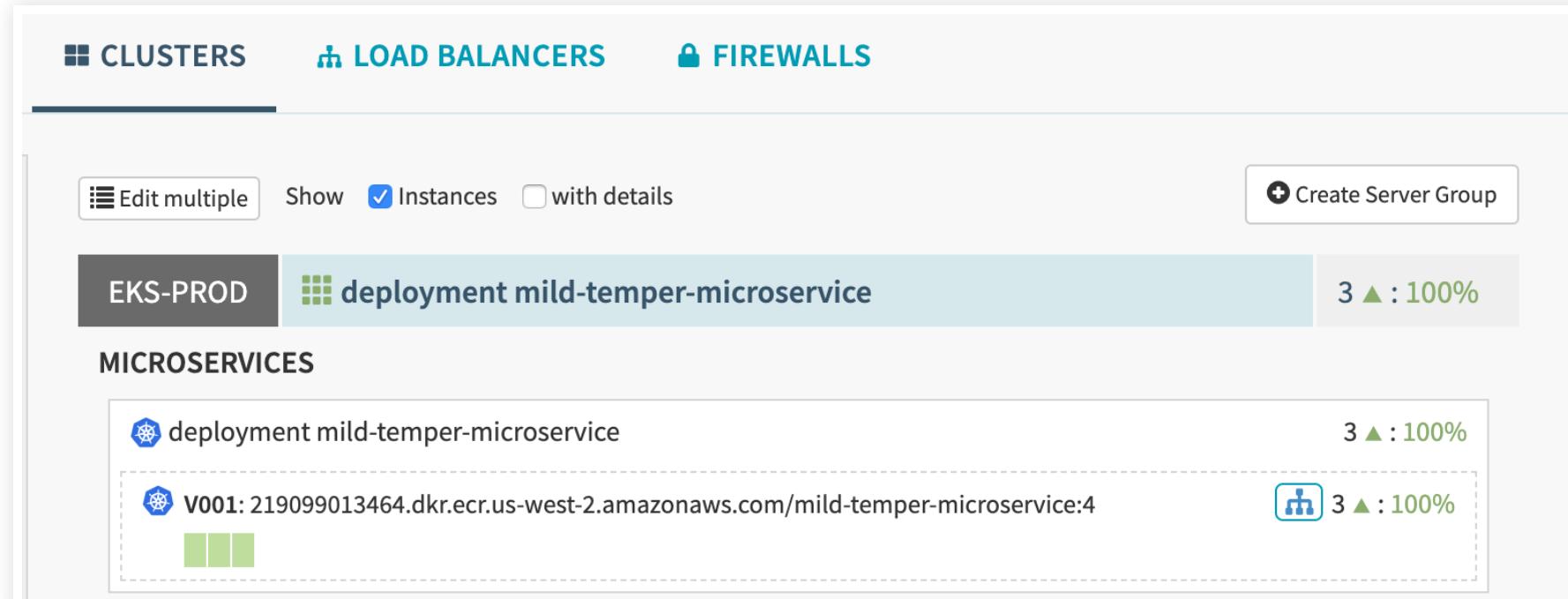
# APPLICATIONS

- Represents the service which you are going to deploy using Spinnaker
- It is the center of your universe in regards to your application
- Contains all configuration for that service, and all the infrastructure on which it will run
- You will typically create a different application for each service

# INSIDE THE APPLICATION

- Firewalls
- Load Balancers
- Clusters of Services or Deployments
- Server Groups
- Canary Configurations

- Does not map to a Kubernetes Cluster
- It *does* map to a Kubernetes *Deployment*



# SERVER GROUP

- Collection of Instances
- With Basic Configuration Settings
  - Number of Instances
  - Autoscaling Policies
  - Their Metadata

# DEPLOYMENT HIGHLIGHTED



Figure 1. The Deployment is highlighted

# REPLICA SET HIGHLIGHTED

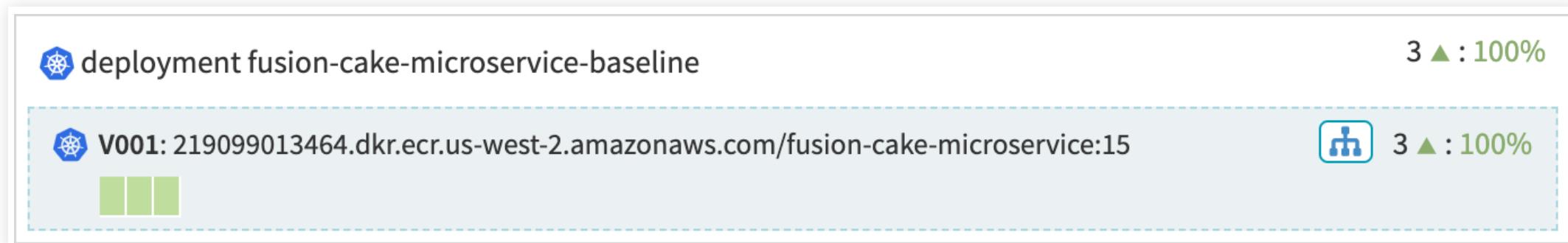
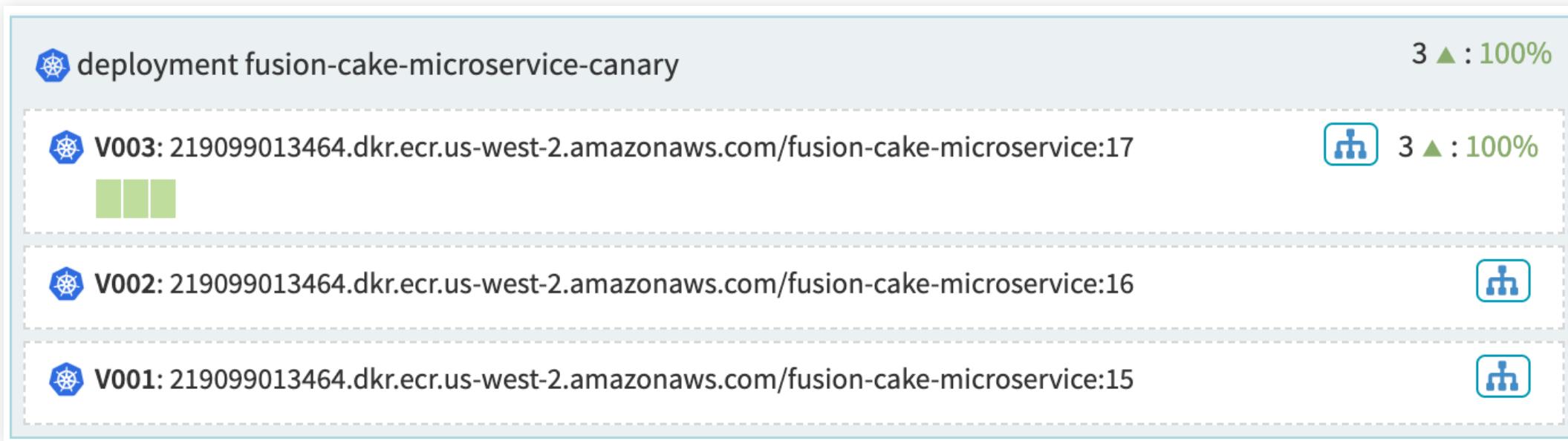


Figure 2. The Replica Set is highlighted

# REPLICA SET HISTORY

- With deployments history of the deployments are maintained
- The following shows a history
- Deployments can be:
  - *rolled back*
  - *deleted*
  - *restarted*
  - *scaled*

# REPLICA SET HISTORY



# LOAD BALANCERS

- Load Balancer is associated with:
  - An ingress protocol
  - Kubernetes Service
- It balances traffic among instances in its Server Groups.
- You can enable health checks for a load balancer
- Flexibility to define health criteria and specify the health check endpoint.

Show  Server Groups ?  Instances ?

 Create Load Balancer

EKS-PROD

ingress mild-temper-ingress



EKS-PROD

service mild-temper-microservice-service



3  : 100%



replicaSet mild-temper-microservice-698848ccf8

EKS-STAGE

ingress mild-temper-ingress



Show  Server Groups  Instances

[Create Load Balancer](#)

EKS-PROD ingress mild-temper-ingress

MICROSERVICES

EKS-PROD service mild-temper-microservice-service

MICROSERVICES 3 ▲ : 100%

replicaSet mild-temper-microservice-698848ccf8

EKS-STAGE ingress mild-temper-ingress

MICROSERVICES

EKS-STAGE service mild-temper-microservice-service

**mild-temper-ingress**

Ingress Actions ▾

**INFORMATION**

Created 2020-03-22 17:42:28 PDT  
Account EKS-STAGE  
Namespace microservices  
Kind Ingress  
Service Type  
Sess. Affinity

**STATUS**

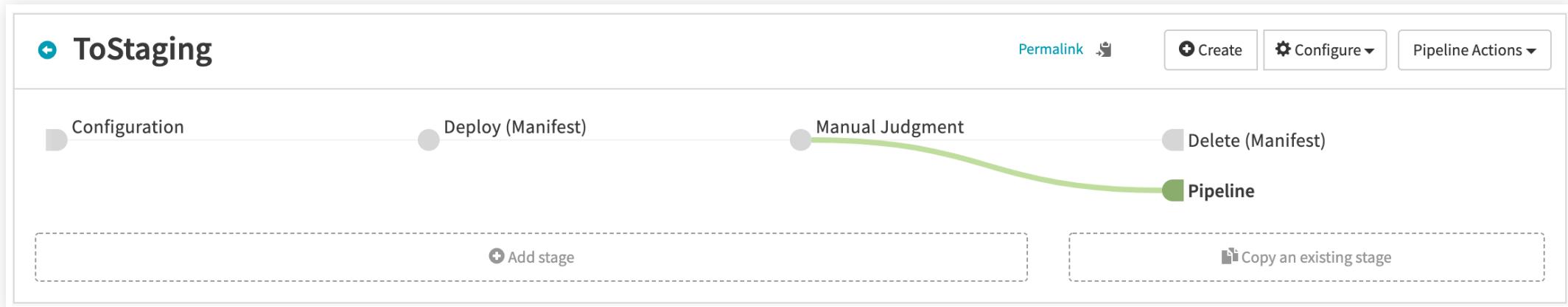
No workloads associated with this Ingress.

**Ingress**  
a9ace9e2a6aef11ea98780edbc813d86-

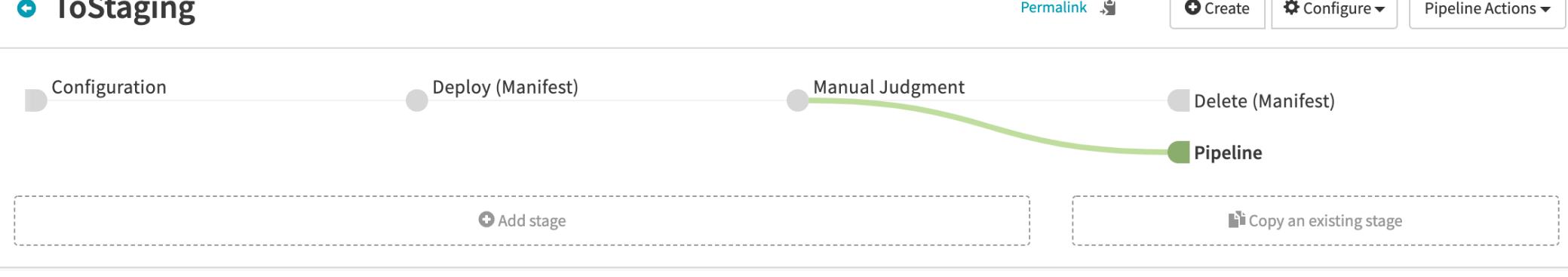
# FIREWALLS

- Firewall defines network traffic access
- Firewall rules defined by an IP range (CIDR) along with a communication protocol (e.g., TCP) and port range
- It is associated with Kubernetes Network Policies

Source: <https://www.spinnaker.io/reference/providers/kubernetes-v2/#networkpolicies>



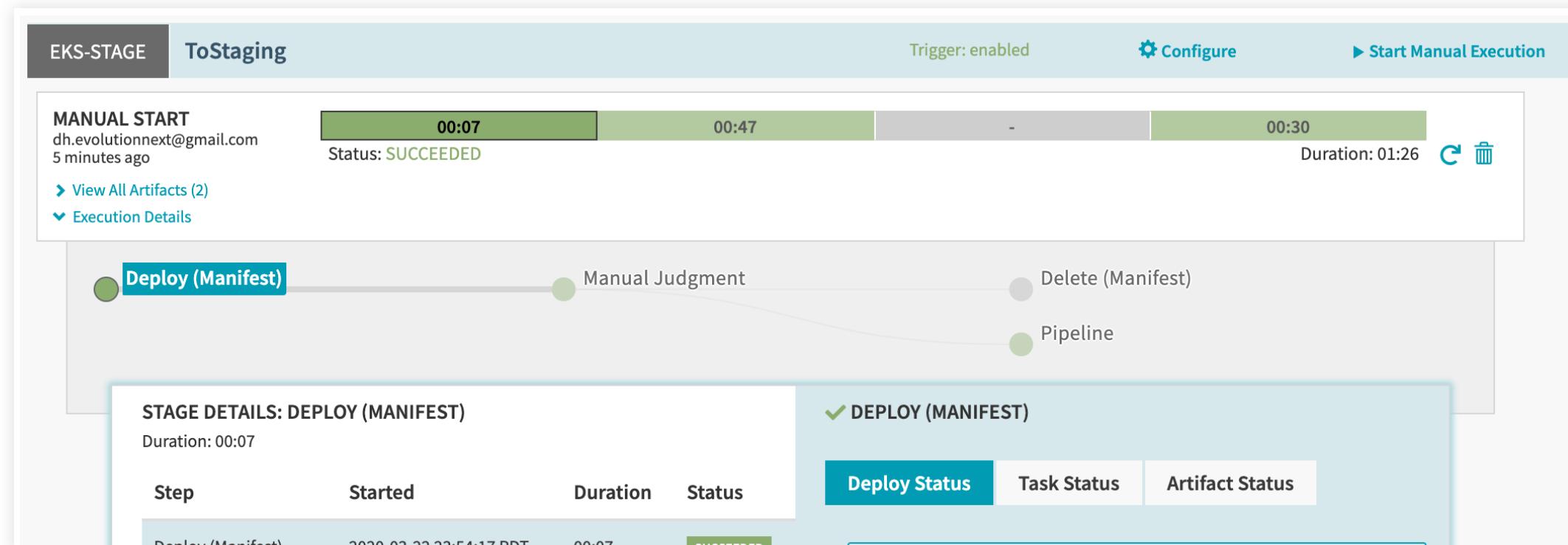
- Key deployment management construct in Spinnaker
- Manage deployments in a consistent, repeatable and safe way.
- Consists of a sequence of actions, known as *stages*
- Possibility to pass parameters from stage to stage along the pipeline using artifacts



- Includes *scaffolding functions* (manual judgment, wait, run Jenkins job) that together precisely defines your runbook for managing your deployments
- Started manually or triggered by an event:
  - git push
  - Docker image upload

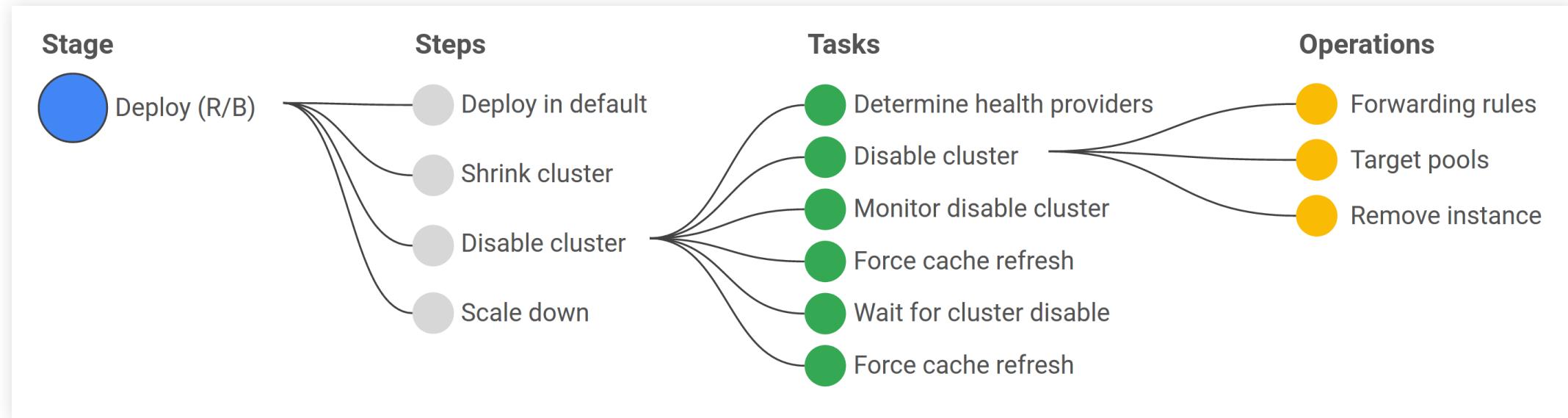
- Spinnaker supports parallel paths of stages, as well as the ability to specify whether multiple instances of a pipeline can be run at once.
- Specify details for a given stage in the sections underneath the pipeline.

- An effective audit log of enforced processes/policies on how you make changes to your deployed applications landscape.

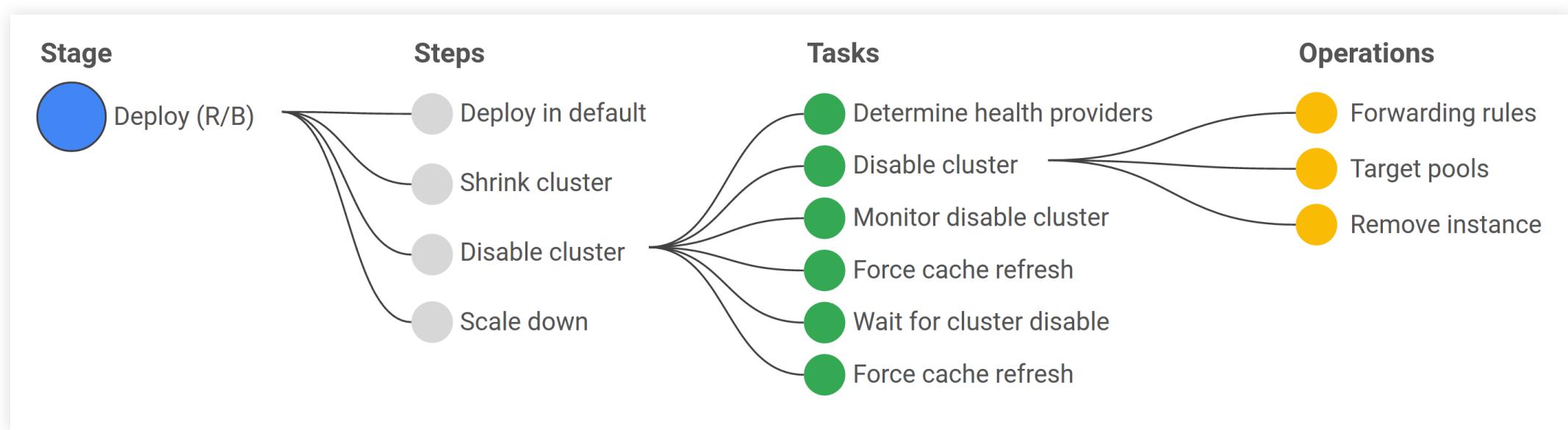




- Atomic building block for a pipeline
- Sequenced to form a pipeline
- Spinnaker provides multiple stages:
  - Deploy
  - Resize



- Automation does not end with orchestrating only the high-level steps of your release process
- Each of these operational steps often corresponds to a sequence of calls to the cloud platform



- The Red/Black Deploy stage in Spinnaker actually entails a sequence of steps
- Each given step is actually a set of tasks that need polling, remediation to ensure requisite state is reached prior to proceeding

<https://github.com/spinnaker/spinnaker/blob/master/applications/microservices/api/docs/deployment-strategies.md>

**mild-temper-microservice**

PIPELINES

INFRASTRUCTURE

TASKS

CONFIG

## Infrastructure

CLUSTERS

LOAD BALANCERS

FIREWALLS

## SEARCH ?

## ACCOUNT

- eks-prod
- eks-stage

## REGION

- microservices

## STACK

- (none)

Show  Instances  with details

EKS-PROD

deployment mild-temper-microservice

3 ▲ : 100%

## MICROSERVICES

deployment mild-temper-microservice

3 ▲ : 100%

V001: 219099013464.dkr.ecr.us-west-2.amazonaws.com/mild-temper-microservice:4

3 ▲ : 100%



EKS-STAGE

deployment mild-temper-microservice

3 ▲ : 100%

## MICROSERVICES

**mild-temper-microservice**

Deployment Actions ▾

## INFORMATION

Created 2020-03-22

21:24:36

PDT

Account EKS-PROD

Namespace microservices

- Spinnaker acts as a single pane of glass from which to manage your global deployments across multiple clouds!
- Contains information related to:
  - Health and status of running environments
  - Metadata around deployments and individual instances
- Also ability to perform ad-hoc actions you can perform on the resources you see such:
  - resize
  - clone
  - disable

# LEFT PANE

The screenshot shows the Spinnaker Infrastructure left pane. The main search bar at the top contains the text "mild-temper-microservice". Below it, there are several filter dropdowns: "ACCOUNT" (eks-prod, eks-stage), "REGION" (microservices), "STACK" ((none)), "DETAIL" ((none)), and "STATUS" (Healthy, Unhealthy, Disabled, Starting). The "CLUSTERS" tab is selected, showing two sections: "EKS-PROD" and "EKS-STAGE". Each section lists a single deployment named "mild-temper-microservice" with a status of "3 ▲ : 100%". A detailed view of the first deployment in the EKS-PROD section is shown on the right, including its creation date (2020-03-22 21:24:36 PDT), account (EKS-PROD), namespace (microservices), kind (deployment), managing (replicaSet), name (mild-temper-microservice-e-698848ccf8), and status (Healthy).

- Filters for viewing your deployed services.
- Search by string match, or narrow by specific attributes of your services, such as:
  - Cloud platform
  - Region
  - Designated environments and stacks.

# MAIN PANE

The screenshot shows the Spinnaker Infrastructure main pane for the 'mild-temper-microservice' application. The left sidebar contains filters for ACCOUNT (eks-prod, eks-stage), REGION (microservices), STACK ((none)), DETAIL ((none)), and STATUS (Healthy, Unhealthy, Disabled, Starting). The main area displays two clusters: EKS-PROD and EKS-STAGE. Each cluster has a 'deployment mild-temper-microservice' entry with a green progress bar indicating 100% completion. A modal window for the EKS-PROD deployment shows detailed information: Created: 2020-03-22 21:24:36 PDT, Account: EKS-PROD, Namespace: microservices, Kind: deployment, Managing: replicaSet, Name: mild-temper-microservice-698848ccf8. The status bar at the bottom shows '100% of 1 instance(s) healthy'.

- Lists the deployed services.
- Green chicklets are individual instances (pods)
- All grouped into Server Groups (a particular deployment)

# RIGHT PANE

The screenshot shows the Spinnaker interface with the 'RIGHT PANE' highlighted. The left sidebar contains navigation links for SPINNAKER, Search, Projects, Applications, Pipeline Templates, and CONFIG. The main center section displays the 'mild-temper-microservice' application under the INFRASTRUCTURE tab, specifically the CLUSTERS section. On the right, a detailed view of the 'mild-temper-microservice' deployment is shown, including its status (3 instances at 100% health), deployment history (v001 to v002), and configuration details like Account (EKS-PROD), Namespace (microservices), Kind (replicaSet), and Managing (mild-temper-microservice). The right pane also includes sections for INFORMATION (creation date, 2020-03-22) and STATUS.

- Provides details for the item currently selected in the center section
- For each item, Spinnaker provides details related to both application-level concerns (Jenkins job details) as well as infrastructure-level concerns (machine type, auto-scaling policies).

- One Application per Microservice

## 2. Define your Infrastructure

- Define infrastructure for each application
- Pipelines deploy services to the server groups you define
- NOTE: Not necessary since you can define infrastructure from pipeline

## 3. Create a Pipeline

- Create all the pipelines needed to deploy the service or services covered by the application

## 4. Run your Pipeline

# CLOUD PROVIDERS

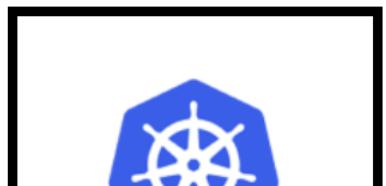
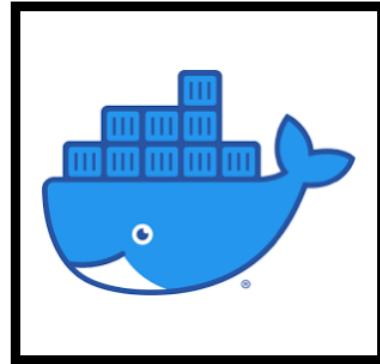
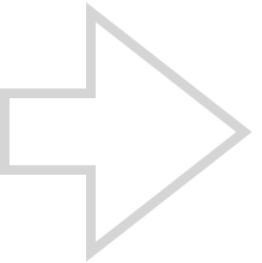
# OVERVIEW

- A Cloud Provider is an interface to a set of virtual resources that Spinnaker has control over.
- Typically, this is a IaaS provider, like AWS, or GCP
- It can also be a PaaS, like App Engine, or a container Orchestrator, like Kubernetes.
- The cloud provider is central to everything you do in Spinnaker.
  - It's where you deploy your Server Groups, the source of your deployable artifacts
  - and the subject of automation via Pipelines.

- An Account is a named credential Spinnaker uses to authenticate against a cloud provider.
- Each provider has slightly different requirements for what format credentials can be in, and what permissions they need to have afforded to them.
- Each Supported Provider has their own way to create an account and register the credentials with **Halyard**.
- Keep in mind that every Provider can have as many accounts added as desired
  - This will allow you to keep your environments (e.g. staging vs. prod):
    - Separate
    - Restrict access to sets of resources using Spinnaker's Authorization mechanisms

- App Engine
- Amazon Web Services
- Azure
- Cloud Foundry
- DC/OS
- Docker v2 Registry (Note: This only acts as a source of images, and does not include support for deploying Docker images)
- Google Compute Engine
- Kubernetes

# DEPLOYING MICROSERVICES



# THE STEPS

1. Setup a repository for a microservice
2. Setup a CI application to run unit integration tests
3. Create an Spinnaker Application
4. Define the Infrastructure
5. Create a Pipeline
6. Run your Pipeline

Source: <https://www.spinnaker.io/guides/user/get-started/>

# CREATE AN APPLICATION

- One Application per Microservice
- This is the root where we will begin
- Note that your application configuration affects what you can do with Spinnaker—you can enable or disable some Spinnaker features.

- An application in Spinnaker is a construct that represents some service that you are going to deploy (typically a microservice).
- It includes:
  - The pipelines that process the service through to deployment in production
  - The infrastructure on which the service is run:
    - Clusters
    - Server Groups
    - Firewalls
    - Load Balancers

The screenshot shows the Spinnaker interface with the 'Applications' tab selected. A search bar at the top right contains the placeholder 'Search applications'. Below it is a table with columns: Name, Created, Updated, Owner, Account(s), and Description. The 'Actions' column contains a 'Create Application' button, which is circled in blue. The table lists seven applications: event, fluentd, heapster, kube, kubernetes, l7, and spin, all owned by 'my-k8s-account'. At the bottom left of the table area is a pagination control with buttons for '«', '<', '1', '>', and '»'.

Name	Created	Updated	Owner	Account(s)	Description
event	-	-		my-k8s-account	
fluentd	-	-		my-k8s-account	
heapster	-	-		my-k8s-account	
kube	-	-		my-k8s-account	
kubernetes	-	-		my-k8s-account	
l7	-	-		my-k8s-account	
spin	-	-		my-k8s-account	

- You can't create a deployment pipeline unless you have an application to put it in.
- Click on the **Applications** tab

- Click Create

### New Application

**Name \*** teama-microservice

 Dashes should only be used in application names when using the Kubernetes v2 provider.

**Owner Email \*** dhinojosa@evolutionnext.com

**Repo Type** github

**Repo Project** https://github.com/dhinojosa/quarkus-microservice.git

**Repo Name** Team A Microservice

**Description** First Exercise! 

**Instance Health**

Consider only cloud provider health when executing tasks 

Show health override option for each operation 

Field	Required	Description
Name	Yes	A unique name to identify this application.
Owner Email	Yes	The email address of the owner of this application, within your installation of Spinnaker.
Repo type	No	The platform hosting the code repository for this application. Stash, Bitbucket, or GitHub.
Description	No	Use this text field to describe the application, if necessary.
Consider only cloud provider	Bool, default=no	If enabled, instance status as reported by the cloud provider is considered sufficient to determine task completion. When disabled, tasks need health

# APPLICATION ATTRIBUTES CONTINUED

Field	Required	Description
Show health override option	Bool, default=no	If enabled, users can toggle previous option per task.
Instance port	No	This field is used to generate links from Spinnaker instance details to a running instance. The instance port can be used or overridden for specific links configured for your application (via the Config screen).
Enable restarting running pipelines	Bool, default=no	If enabled, users can restart pipeline stages while a pipeline is still running. This behavior is not recommended.

If you need to go back and edit your configuration you can hit the config button on the upper right hand side



The Configuration has some of the following sections:

- Edit Notifications
- Features
- Links
- Traffic Guards
- Custom Banners
- Delete Application

Spinnaker can send notification for any application and are optional

**Click Add Notification Preferences**

**Edit Notification**

**Notify via** Email

**Email Address** person\_of\_interest@example.com

**CC Address** innocent\_bystander@example.com

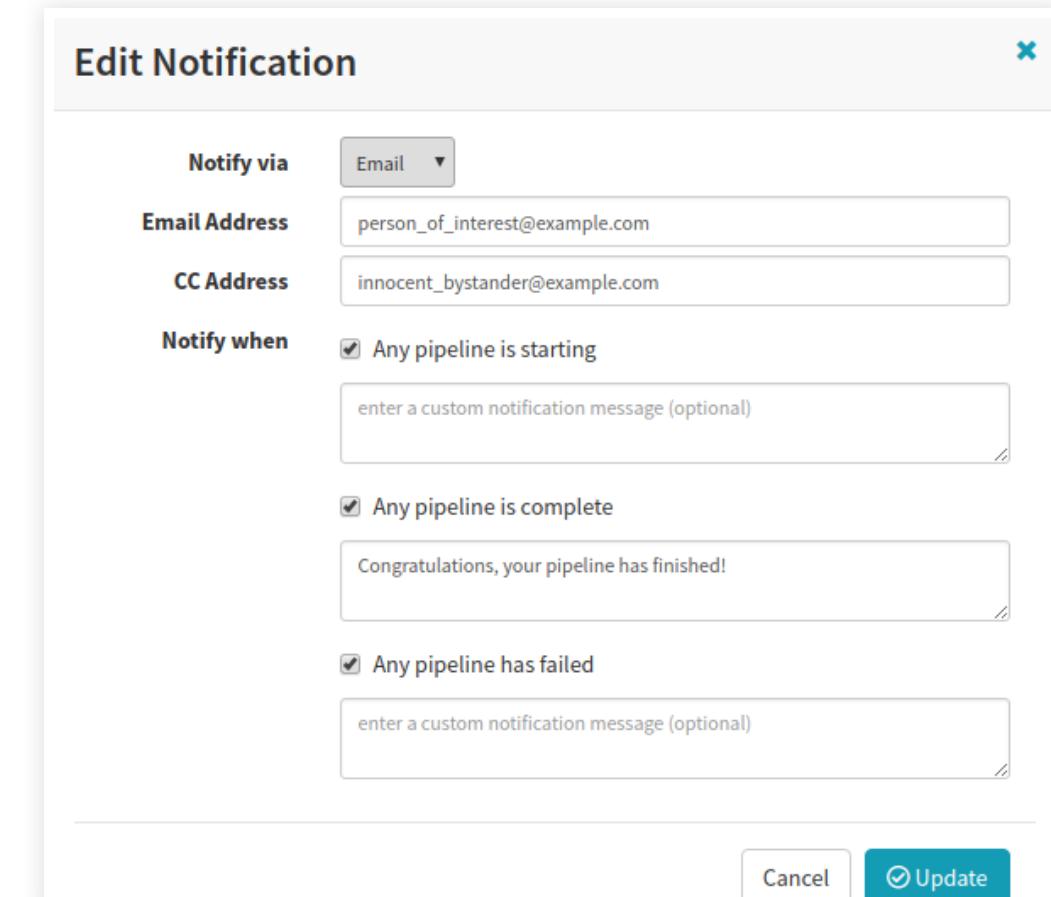
**Notify when**

Any pipeline is starting  
enter a custom notification message (optional)

Any pipeline is complete  
Congratulations, your pipeline has finished!

Any pipeline has failed  
enter a custom notification message (optional)

**Cancel** **Update**



- Select the notification medium:
  - Email
  - SMS
  - Slack

**Edit Notification**

**Notify via**

**Email Address**

**CC Address**

**Notify when**

Any pipeline is starting

Any pipeline is complete

Any pipeline has failed

- Provide the information for the type you chose:
  - Email address
  - Phone number
  - Slack Members

### Edit Notification

Notify via

Email Address

CC Address

Notify when

Any pipeline is starting

Any pipeline is complete

Any pipeline has failed

For each event, enter any custom text you want included in the notification.

- A pipeline started
- A pipeline finished
- A pipeline failed

Edit Notification ✖

Notify via Email

Email Address

CC Address

Notify when

Any pipeline is starting

Any pipeline is complete

Any pipeline has failed

---

Cancel Update

**1. Click Update**

**2. Repeat these steps to add more notifications, if you want.**

**3. Click **Edit** next to any existing notification to change its preferences, or click **Remove** to delete it.**

**Edit Notification**

**Notify via** Email

**Email Address** person\_of\_interest@example.com

**CC Address** innocent\_bystander@example.com

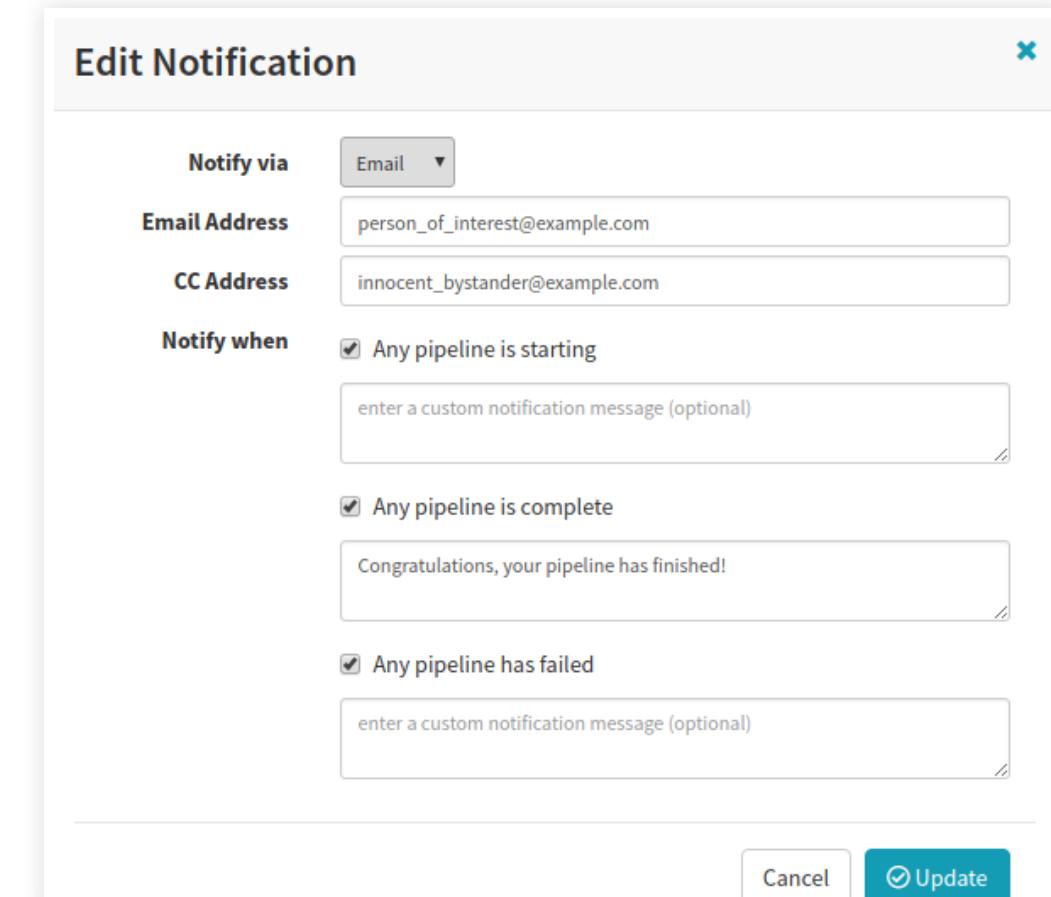
**Notify when**

Any pipeline is starting  
enter a custom notification message (optional)

Any pipeline is complete  
Congratulations, your pipeline has finished!

Any pipeline has failed  
enter a custom notification message (optional)

**Cancel** **Update**



In order to not allow Spinnaker from annoying you about resources you don't use you can turn them off

## Features

If you don't need or want certain features for your application, you can disable them here.

Disabling a feature only changes the display in Spinnaker - it won't delete any actual data.

**Pipelines**

Orchestrated deployment management

**Clusters**

Collections of server groups or jobs

**Load Balancers**

Traffic distribution management between servers

**Firewalls**

Network traffic access management

**Canary**

Canary analysis configuration and reporting

information related to the instance, like logs, health, and so on.

- It currently doesn't work with Kubernetes resources unless exposed from cluster

## Links

Links appear in the instance details panel and provide a shortcut to common features, such as logs, health, etc.

Links paths are templates - you can customize them using a number of field attributes available on the instance.

[Check out the guide](#) for examples and to see what fields are available for your cloud provider.

[Edit as JSON](#)

Section Heading	Name	URL
<b>Links</b>	Label, e.g. Health	Path, e.g. /health, :7002/alt-port-link

1. In the **Links** section, click **Add Section**
  2. Provide text for the new **Section heading**.
  3. In the **Links** fields, type the text that you want to display, and the path to the link target.
  4. You can reference these specific instance attributes by wrapping them in curly braces. For example, to include the region where the instance lives, for a GCE or AWS instance, use `{region}`. These are for the path only, not the link display text.
  5. Click **Add Link** for each additional link you want displayed in this section.
  6. Click **Add Section** for any additional sections.
  7. Click **Revert** to clear your work in progress.
8. This does not remove any saved custom instance links.

## Traffic Guards

Traffic Guards allow you to specify critical clusters that should always have active instances. If a user or process tries to delete, disable, or resize the server group, Spinnaker will verify the operation will not leave the cluster with no active instances, and fail the operation if it would.

Enabled	Account	Region <small>?</small>	Stack <small>?</small>	Detail <small>?</small>	Matched clusters
<input checked="" type="checkbox"/>	<div><input checked="" type="checkbox"/> Select... eks-prod eks-stage</div>	(select account)			(no matches) <span style="color: red;">!</span>
<a href="#">+ Add Traffic Guard</a>					
			<a href="#">⚡ Revert</a>	<a href="#" style="background-color: #009640; color: white; padding: 5px 10px; border-radius: 5px;">✔ Save Changes</a>	

- A traffic guard is a cluster which you have designated as always having at least one active instance.
- With traffic guard enabled, if a user or process tries to delete, disable, or resize the server

- Click **Add Traffic Guard**
- Set the following values as seen in table below
- Click **Save Changes** to apply your new traffic guard, or changes to an existing one

Field	Required?	Description
Account	Yes	The account for which you're setting up the traffic guard
Region	Yes	The applicable region from the list, or * for all
Stack	No	The stack to which to apply this traffic guard. Leave blank to apply the guard only to a cluster that has no stack.

# CUSTOM BANNERS

## Custom Banners

Custom Banners allow you to specify application-specific headers that will appear above the main Spinnaker navigation bar.

Enabled	Text	Text Color	Background
<input type="checkbox"/>	Your custom banner text <small>Markdown is okay ⓘ</small>	<input type="button" value=""/>	<input type="button" value=""/> 
<b>Preview</b> Your custom banner text			
<b>Add banner</b>			

 Revert

 Save Changes

# CUSTOM BANNER PROCESS

You can create simple text based banners

1. Click **Add Banner**
2. Click **Enable** to enable the banner
3. Select a text color for the foreground color
4. Select a background color for the background color

# DEFINE THE INFRASTRUCTURE THE SERVICE WILL RUN

- You define infrastructure for each application.
- Your pipelines deploys services to the server groups you define.
- This means that production environments can be deployed without a pipeline.
- This will provide you with the initial production environment

**i** this step is not a prerequisite for creating pipelines. In fact, you can use pipelines to create infrastructure.

# PIPELINES

- How we control and deploy our application
- Series of stages that can be combined in any order
- Can be automatic or manual
- Triggered by a wide arrange of inputs

- Name your pipeline

- Then add stages

The screenshot shows the Spinnaker interface for managing pipelines. At the top, there's a navigation bar with 'SPINNAKER', 'Search', 'Projects', and 'Applications'. On the right side of the top bar are 'Search', 'What's New', and 'Help'. Below the top bar, the main header includes the project name 'example', tabs for 'PIPELINES', 'INFRASTRUCTURE', 'TASKS', and 'CONFIG', and a search bar.

The 'PIPELINES' tab is active, showing a search bar with placeholder text 'example pipeline'. Below the search bar, it says 'No executions found matching the selected filters.' To the left of the search bar is a sidebar with sections for 'SEARCH', 'PIPELINES' (containing 'example pipeline'), and 'STATUS' (with options for 'Running', 'Terminal', 'Succeeded', 'Not Started', and 'Cancelled').

At the top right of the main content area, there are buttons for 'Create' (which is highlighted with a red box), 'Configure', and 'Start Manual Execution'. There are also dropdowns for 'Group by Pipeline' and 'Show 2 executions per pipeline'.

# CONFIGURATION STAGE

- The initial stage that you are provided is the configuration stage
- It is where we define:
  - *expected artifacts*
  - *triggers*
  - *parameters*
  - *notifications*

SPINNAKER    Search    Projects    Applications

Search  What's New Help ▾

 example  PIPELINES INFRASTRUCTURE TASKS CONFIG

**example pipeline**

Configuration  

**CONCURRENT EXECUTIONS** **Concurrent Executions**

Disable concurrent pipeline executions (only run one at a time).  
 Do not automatically cancel pipelines waiting in queue.

**AUTOMATED TRIGGERS** **Automated Triggers**

You don't have any triggers configured for example pipeline.



# ADDING STAGE PROCESS

- The first step in any pipeline is **Configuration**, where you can set up pipeline triggers and parameters.
- Each stage specifies an action the pipeline will take once it's configured.
- The steps include
  - Select Add stage from your pipeline configuration screen.
  - Set the stage type using the drop-down menu.

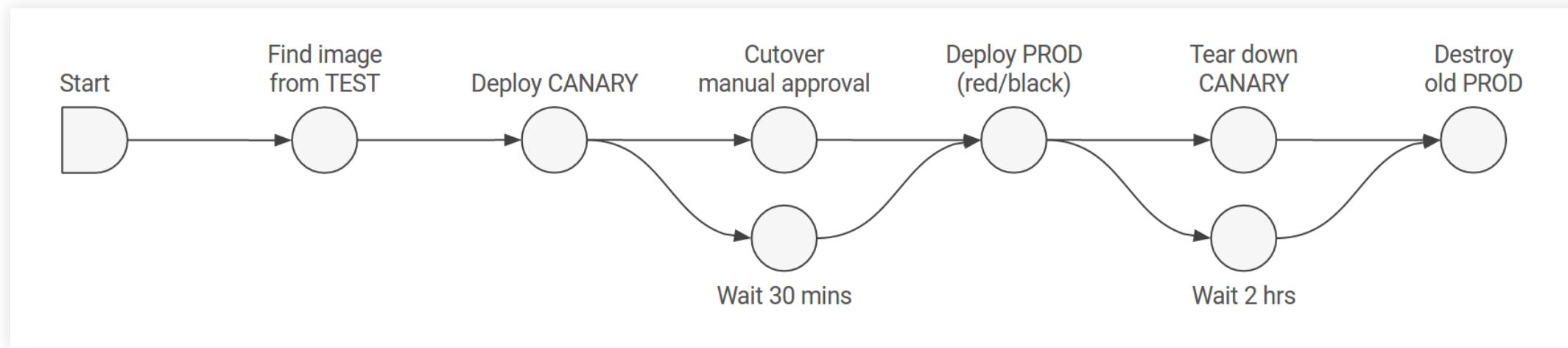
If this is the first stage in your pipeline, make sure that this stage depends on the desired upstream stage(s) using the Depends on field.

The screenshot shows the Spinnaker interface for managing pipelines. At the top, there's a navigation bar with 'SPINNAKER', 'Search', 'Projects', and 'Applications' tabs. To the right are 'Search', 'What's New', and 'Help' options. Below the navigation is a secondary header with 'example', 'PIPLINES', 'INFRASTRUCTURE', 'TASKS', and 'CONFIG' tabs. The main content area is titled 'example pipeline'. It displays a flow diagram with three stages: 'Configuration', 'Bake', and 'Deploy'. A green line connects 'Configuration' to 'Bake', and another green line connects 'Bake' to 'Deploy'. Below the diagram are two dashed boxes: one labeled 'Add stage' and another labeled 'Copy an existing stage'. Underneath the flow diagram, there's a detailed configuration panel for the 'Deploy' stage. The 'Stage type' is set to 'Deploy', and the 'Stage Name' is 'Deploy'. The 'Depends On' field contains the value 'Bake', which is highlighted with a red rectangular box. To the right of this field are buttons for 'Remove stage' and 'Edit stage as JSON'. At the bottom of the configuration panel, there are tabs for 'DEPLOY CONFIGURATION' (which is selected), 'EXECUTION OPTIONS', and a 'Deploy Configuration' section containing several dropdown menus.

Care to, particularly if you don't want to redo the build, you can click on **Start Manual Execution**

The screenshot shows the Spinnaker interface for the 'example' project. The top navigation bar includes 'SPINNAKER', 'Search', 'Projects', 'Applications', a search bar, 'What's New', and 'Help'. Below the navigation is a secondary header with tabs for 'PIPELINES' (selected), 'INFRASTRUCTURE', and 'TASKS'. On the left, a sidebar contains sections for 'SEARCH', 'PIPELINES' (with an 'example pipeline' checkbox), and 'STATUS' (with checkboxes for Running, Terminal, Succeeded, Not Started, Canceled, and Stopped). The main content area displays the 'example pipeline' with a message: 'No executions found matching the selected filters.' To the right of the pipeline name are 'Configure' and 'Start Manual Execution' buttons. The 'Start Manual Execution' button is highlighted with a red box.

# APPLICATION DEPLOYMENT



- From the Pipelines tab, click **Configure** to modify an existing pipeline.
- Click Pipeline actions in the upper right corner, and select **Disable**.

The screenshot shows the Spinnaker interface for managing pipelines. At the top, there's a navigation bar with 'SPINNAKER', 'Search', 'Projects', 'Applications', a search bar, 'What's New', and 'Help'. Below the navigation is a header with tabs for 'example', 'PIPLINES' (which is selected), 'INFRASTRUCTURE', 'TASKS', and 'CONFIG'. The main content area is titled 'example pipeline' and shows a pipeline configuration with stages 'Configuration' and 'Bake'. On the left, there are tabs for 'CONCURRENT EXECUTIONS' (selected), 'AUTOMATED TRIGGERS', 'PARAMETERS', 'NOTIFICATIONS', and 'DESCRIPTION'. Under 'CONCURRENT EXECUTIONS', there are two sections: 'Concurrent Executions' with checkboxes for 'Disable concurrent pipeline executions (only run one at a time)' (checked) and 'Do not automatically cancel pipelines waiting in queue.', and 'Automated Triggers'. In the top right corner of the pipeline header, there's a 'Pipeline Actions' dropdown menu with a red box around it. The menu contains the following options: 'Rename', 'Delete', 'Disable' (which is also highlighted with a red box), 'Lock', 'Edit as JSON', and 'No version history found'.

From the Pipelines tab, click **Configure** to modify an existing pipeline.

- Click **Pipeline actions** in the upper right corner, and select **Delete**.

The screenshot shows the Spinnaker interface for managing pipelines. At the top, there's a navigation bar with 'SPINNAKER', 'Search', 'Projects', 'Applications', a search bar, 'What's New', and 'Help'. Below the navigation is a header with tabs: 'example' (highlighted), 'PIPELINES' (selected), 'INFRASTRUCTURE', 'TASKS', and 'CONFIG'. The main content area is titled 'example pipeline'. It shows a flow diagram with stages: 'Configuration' (green) and 'Bake' (grey). Below the diagram are sections for 'CONCURRENT EXECUTIONS' (selected), 'AUTOMATED TRIGGERS', 'PARAMETERS', 'NOTIFICATIONS', and 'DESCRIPTION'. The 'CONCURRENT EXECUTIONS' section contains two checkboxes: 'Disable concurrent pipeline executions (only run one at a time)' (checked) and 'Do not automatically cancel pipelines waiting in queue'. A 'Pipeline Actions' dropdown menu is open on the right, containing options: 'Rename', 'Delete', 'Disable', 'Lock', 'Edit as JSON', and 'No version history found'. The 'Delete' option is highlighted with a red box.

The screenshot shows the Spinnaker interface for managing pipelines. At the top, there's a navigation bar with 'SPINNAKER' and links for 'Search', 'Projects', and 'Applications'. On the right side of the top bar are 'Search', 'What's New', and 'Help' options. Below the top bar, the main header includes the project name 'example', pipeline navigation ('PIPELINES', 'INFRASTRUCTURE', 'TASKS'), and a 'CONFIG' tab.

The main content area displays the 'example pipeline'. It features a visual pipeline diagram with a single stage named 'Bake'. Below the diagram are sections for 'CONCURRENT EXECUTIONS' and 'AUTOMATED TRIGGERS'. The 'CONCURRENT EXECUTIONS' section contains two checkboxes: one checked for 'Disable concurrent pipeline executions (only run one at a time)' and another unchecked for 'Do not automatically cancel pipelines waiting in queue'. The 'AUTOMATED TRIGGERS' section states 'You don't have any triggers configured for example pipeline.' and includes an 'Add Trigger' button.

A context menu is open on the right side of the pipeline diagram, with the 'Pipeline Actions' option highlighted by a red box. The menu lists several actions: 'Rename', 'Delete', 'Disable', 'Lock', 'Edit as JSON', and 'No version history found'.

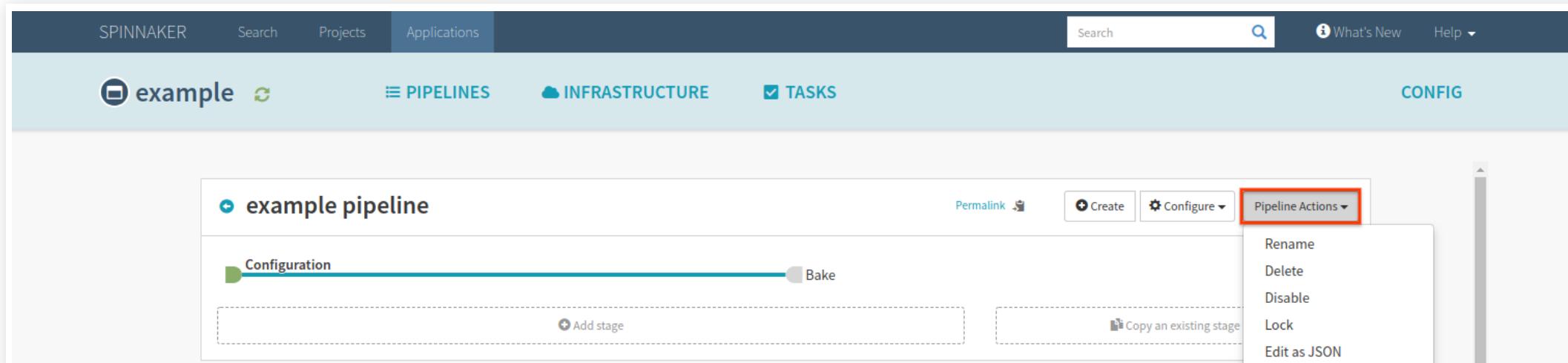
# EDITING AS JSON PROCESS

- You can also edit your pipeline in JSON
- This is fraught with danger if you get it wrong
- It can be helpful if things go wrong already
  - Perhaps the UI is not updating the backend correctly
  - You wish to diagnose issues
- When you use the **Edit as JSON** feature, you are directly editing the payload.
- Edit as JSON allows you to set pipeline fields or properties not exposed by the UI.

- You can use revision history to diff two versions of a pipeline or to restore an older version of a pipeline.

1. From the **Pipelines** tab, click **Configure** to modify an existing pipeline

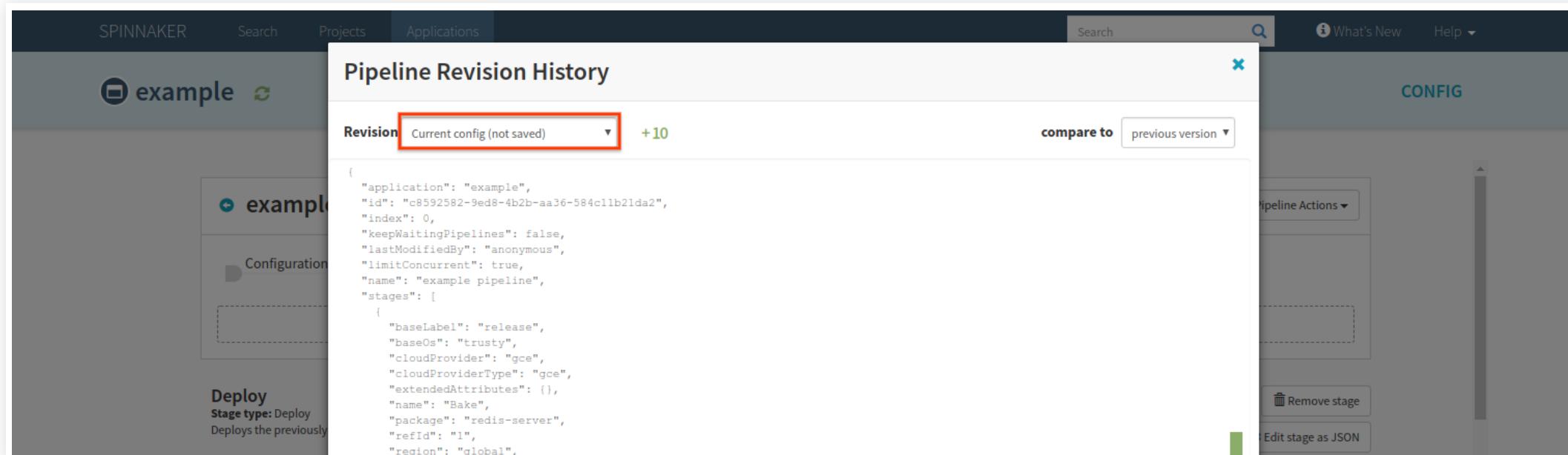
2. Click **Pipeline Actions** in the upper right corner, and select **View revision history**



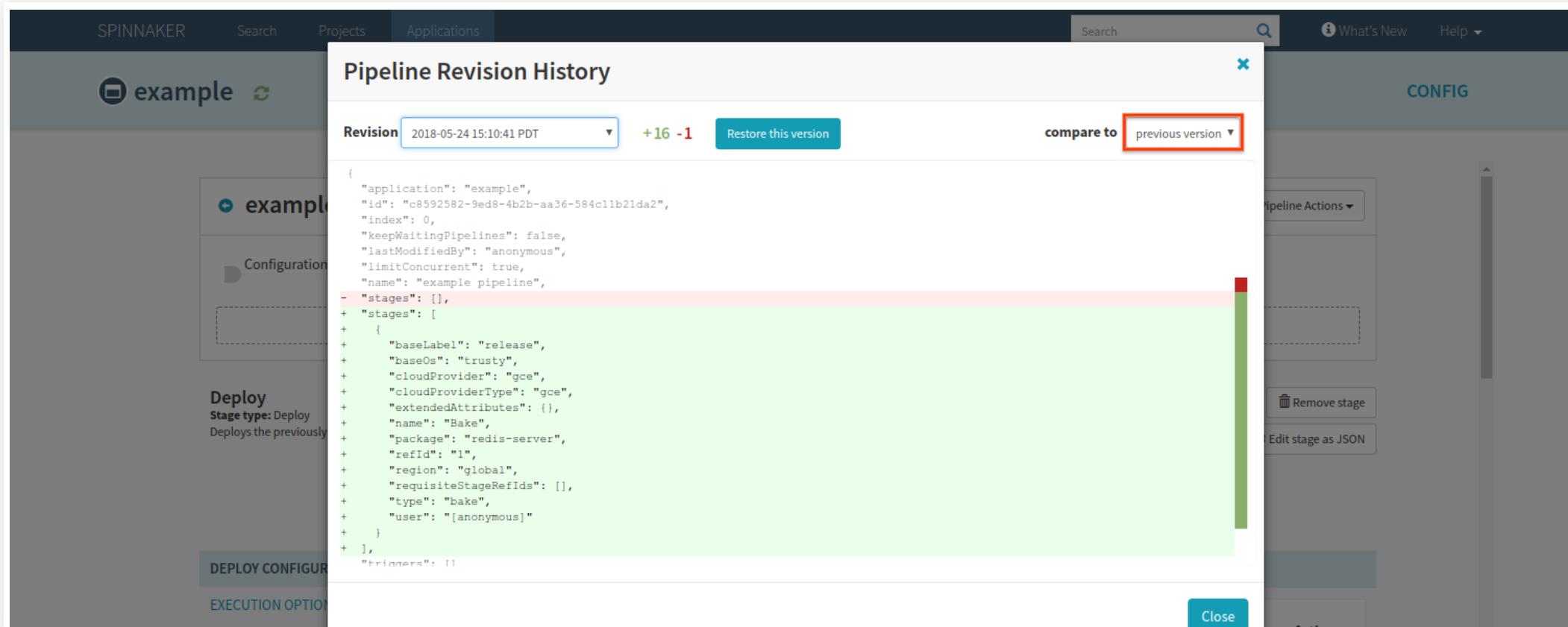
pipeline.

## 1. View the version history

## 2. Select a version from the Revision drop-down.



You can also compare a review to either the current revision or the previous one



If you select a different version, you can revert to that same version

The screenshot shows the Spinnaker Pipeline Revision History dialog. At the top, there's a revision dropdown set to "2018-05-24 15:10:41 PDT" with a green "+16" and a red "-1" next to it, and a "Restore this version" button highlighted with a red box. Below the button is a diff view of a JSON pipeline configuration. The configuration includes fields like "application", "id", "index", "keepWaitingPipelines", "lastModifiedBy", "limitConcurrent", "name", and "stages". The "stages" field is expanded, showing a list of stages with attributes such as "baseLabel", "baseOs", "cloudProvider", "cloudProviderType", "extendedAttributes", "name", "package", "refId", "region", "requisiteStageRefIds", "type", and "user". A sidebar on the right contains "Pipeline Actions" like "Remove stage" and "Edit stage as JSON", and an "Actions" section.

```
{
  "application": "example",
  "id": "c8592582-9ed8-4b2b-aa36-584c11b21da2",
  "index": 0,
  "keepWaitingPipelines": false,
  "lastModifiedBy": "anonymous",
  "limitConcurrent": true,
  "name": "example pipeline",
  - "stages": [],
  + "stages": [
    +
      "baseLabel": "release",
      "baseOs": "trusty",
      "cloudProvider": "gce",
      "cloudProviderType": "gce",
      "extendedAttributes": {},
      "name": "Bake",
      "package": "redis-server",
      "refId": "1",
      "region": "global",
      "requisiteStageRefIds": [],
      "type": "bake",
      "user": "[anonymous]"
    +
  ],
  "triggers": []
}
```

2. From the Pipelines tab, click **Configure** to modify an existing pipeline.

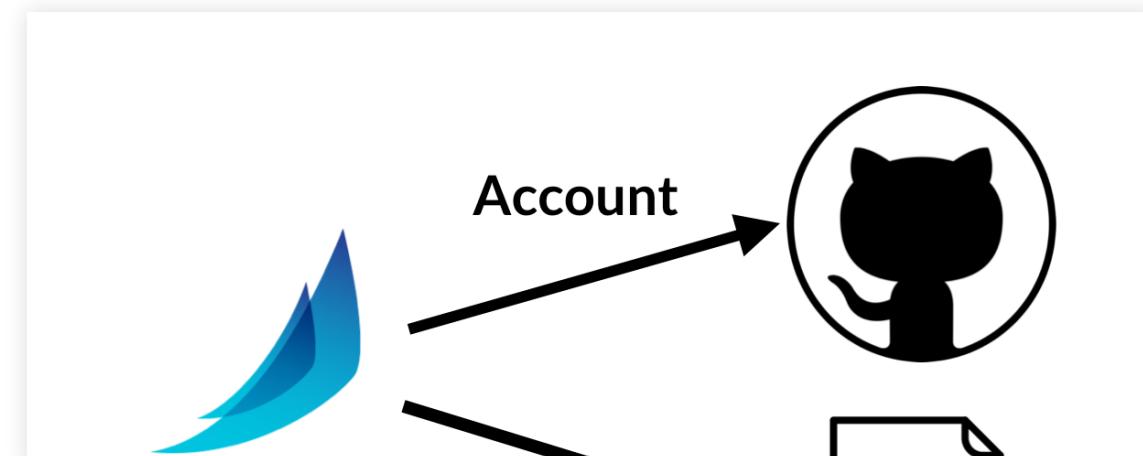
## 2. Click Pipeline actions in the upper right corner, and select Rename

The screenshot shows the Spinnaker interface for managing pipelines. At the top, there's a navigation bar with 'SPINNAKER', 'Search', 'Projects', 'Applications', 'What's New', and 'Help'. Below the navigation is a header with tabs: 'example' (selected), 'PIPLINES' (selected), 'INFRASTRUCTURE', 'TASKS', and 'CONFIG'. The main content area is titled 'example pipeline' and shows a flow diagram with stages: 'Configuration' (green) and 'Bake' (grey). There are buttons for 'Add stage' and 'Copy an existing stage'. On the left, there are sections for 'CONCURRENT EXECUTIONS' (selected), 'AUTOMATED TRIGGERS', 'PARAMETERS', 'NOTIFICATIONS', and 'DESCRIPTION'. In the 'CONCURRENT EXECUTIONS' section, there are checkboxes for 'Disable concurrent pipeline executions (only run one at a time)' (checked) and 'Do not automatically cancel pipelines waiting in queue' (unchecked). A 'Pipeline Actions' dropdown menu is open at the top right, with a red box highlighting it. The menu contains the following options: 'Rename', 'Delete', 'Disable', 'Lock', 'Edit as JSON', and 'No version history found'.

# LAB 1: SPINNAKER INTRODUCTION

# ARTIFACTS

- A Docker image
- A file stored in GitHub
- An Amazon Machine Image (AMI)
- A binary blob in Amazon S3, Google Cloud Storage, etc.



# ARTIFACT FRUSTRATION

Artifacts are one of the most frustrating aspects to Spinnaker

```
Failed on startup: Unmatched expected artifact  
ExpectedArtifact(matchArtifact=Artifact(type=s3/object,  
customKind=false, name=s3://XXXXXXXXXXXX-alpha.191%2B926bb3b3.tgz,  
version=null, location=null, reference=null, metadata={id=01317db7-1e30-  
4c02-a302-826449d83f94}, artifactAccount=f5cs-spin, provenance=null,  
uuid=null), usePriorArtifact=false, useDefaultArtifact=false,  
defaultArtifact=Artifact(type=null, customKind=true, name=null,  
version=null, location=null, reference=null, metadata={id=ca86ca54-6719-  
471e-b10e-fb4a6f20e85d}, artifactAccount=null, provenance=null,  
uuid=null), id=0730e728-e19a-43af-b9e5-9856332e939d, boundArtifact=null)  
could not be resolved.
```

# WHAT IS THE "ADDRESS" OF OUR ARTIFACTS?

- URI of the location of our artifact
- Provenance information about the commit that triggered the resource



*Provenance def:* - The place of origin or earliest known history of something

- To allow more provenance information, a JSON is used to describe extras like:
  - name
  - version
  - location
- This is called *artifact decoration*
- Every Spinnaker artifact follows this specification whether:
  - Supplied to a pipeline
  - Accessed within a pipeline

## following Spinnaker artifact:

```
{  
  "type": "gcs/object",  
  "reference": "gs://bucket/file.json#135028134000",  
  "name": "gs://bucket/file.json",  
  "version": "135028134000"  
  "location": "us-central1"  
}
```

## Another example

```
{  
  "type": "docker/image",  
  "reference": "gcr.io/project/image@sha256:29fee8e284",  
  "name": "gcr.io/project/image"
```

type	How the external resource is classified. (This allows for easy distinction between Docker images and Debian packages, for example).	
reference	The URI used to fetch the resource.	
artifactAccount	The Spinnaker artifact account that has permission to fetch the resource.	
version	The version of the resource. (By convention, version should only be compared between artifacts of the same type and name.)	Optional.

# SPINNAKER ARTIFACT FIELDS CONTINUED

metadata	Arbitrary key / value metadata pertaining to the resource.  (This can be useful for scripting within pipeline stages.)	Optional.
location	The region, zone, or namespace of the resource. (This does not add information to the URL, but makes multi-regional deployments easier to configure.)	Optional.
uuid	Used for tracing the artifact within Spinnaker.	Assigned by Spinnaker.

## EXPECTED ARTIFACTS

- Expects a particular artifact to be available
- Declarable within a pipeline trigger or stage
- Spinnaker compares an incoming artifact (for example, a manifest file stored in GitHub) to the expected artifact (for example, a manifest with the file path `path/to/my/manifest.yml`)
- If there is a match it is *bound* to that artifact and used

## Expected Artifacts

Declare artifacts your pipeline expects during execution in this section. 

Match against 

 Remove artifact

**Account**

 abollapr-account



**File path** 

manifests/deploy-microservice.yml

---

**Display name**

soul-source-deploy-microservice.yml

---

If missing 

**Use Prior Execution**

**Use Default Artifact**

## MATCHED ARTIFACTS

- Used to disambiguate between similar artifacts coming from the same account
  - e.g. Different `yaml` files from the same repository
- Filter by fields in the artifact format against which to compare in the incoming artifact
- Used to specify that the trigger should begin pipeline execution only if the incoming artifact matches the parameters that you provided

# PRIOR EXECUTION AND DEFAULT ARTIFACTS

If missing ⓘ

**Use Prior Execution**

**Use Default Artifact**

Default artifact ⓘ

<b>Account</b>	 abollapr-account
<b>Content URL</b> ⓘ	<a href="https://api.github.com/repos/abollapr/quarkus-microservice/contents/manifests/de">https://api.github.com/repos/abollapr/quarkus-microservice/contents/manifests/de</a>
<b>Commit/Branch</b> ⓘ	master

# PRIOR EXECUTION AND DEFAULT ARTIFACTS FIELDS

- If Missing
  - Provide fallback behavior for the expected artifact in case the trigger doesn't find the desired artifact.
- Use prior execution
  - Spinnaker will fall back to the artifact used in the last execution.
  - If you enable the **Use default artifact** Spinnaker will use a default artifact,
  - Allows you to provide fallback behavior for the first time a trigger is used, when no previous execution exists

## REVIEW OF ARTIFACTS

- Expected - An artifact that a trigger or stage expects to have
- Bound - An artifact that has been fulfilled
- Matched - Enter fields to disambiguate similar artifacts

## ARTIFACTS IN PIPELINES DEFINED

- An artifact arrives in a pipeline execution either from an external trigger (for example, a Docker image pushed to a registry)
- Getting fetched by a stage. That artifact is then consumed by downstream stages based on predefined behavior.

## EXPECTED ARTIFACTS

- Enable a stage to bind an artifact from:
  - Another pipeline execution
  - Stage output
  - Running environment

have present before it begins a pipeline execution!

## Automated Triggers

Type

Docker Registry

x ▾

 Remove trigger

Define Image ID

Select from list

Registry Name

my-ecr-registry

x ▾



Organization

No organization

Image

soul-source-microservice

x ▾

Tag 

Artifact from Execution" stage.

- All that's required is the pipeline ID whose execution history to search, and an expected artifact to bind.
- This will be particularly useful for canary attainment of baseline

Find Artifact From Execution Configuration

Pipeline Selector

Application	cluster
Pipeline	find

Execution Options



# ARTIFACTS PASSED BETWEEN PIPELINES

- Artifacts can be passed between pipelines.
- Two cases:
  - Pipeline B is triggered by Pipeline A - Pipeline B is triggered by Pipeline A, therefore Pipeline B will have Pipeline A's artifacts
  - Pipeline A triggers Pipeline B - Pipeline A is triggers Pipeline B, therefore Pipeline B will have Pipeline A's artifacts

Stages can be configured to ‘Produce’ artifacts if they expose the following Stage configuration:

The screenshot shows a configuration dialog for 'Produces Artifacts'. At the top, a blue header bar contains the title 'Produces Artifacts'. Below the header, there are several configuration sections:

- Match against**: A dropdown menu set to 'GCS' with a downward arrow, indicating the artifact type. To its right, the text 'A GCS object.' is displayed.
- Object path**: A text input field containing the value 'gs://manifest/baked-manifest.yml'.
- If missing**: A section with the label 'Use Default Artifact' followed by an empty checkbox input field.
- Add Artifact**: A dashed rectangular button with a plus sign and the text 'Add Artifact'.

# WAYS TO PRODUCE ARTIFACTS

1. To bind artifacts injected into the stage context - If your stage emits artifacts (such as a “Deploy (Manifest)” stage) into the pipeline context, you can match against these artifacts for downstream stages to consume.
2. To artificially inject artifacts into the stage context - If you are running a stage that does not natively emit artifacts (such as the “Run Job” stage), you can use the default artifact, which always binds to the expected artifact, to be injected into the pipeline each time it is run. Keep in mind: If the matching artifact is empty, it will bind any artifact, and your default artifact will not be used.

- i** If you are configuring your stages using JSON, the expected artifacts are placed in a top-level `expectedArtifacts`:  
[ ] list.

- Docker Image
- Embedded Base64
- GCS Object
- S3 Object
- Git Repo
- GitHub File
- GitLab File
- Bitbucket File

## SOURCES

<https://www.spinnaker.io/reference/artifacts-with-artifactsrewrite/>

<https://www.spinnaker.io/reference/artifacts-with-artifactsrewrite/in-kubernetes-v2/>

# LINKING SPINNAKER TO GITHUB

# OVERVIEW

- In order to connect to github, we will need to establish a token
- This will build a connection from Spinnaker to Github
- As users you will create the token
- As administration you will put the token in Spinnaker using Halyard

# CREATING A TOKEN

- Visit your github account at: <https://github.com/settings/tokens>
- Click on the **Generate new token** button

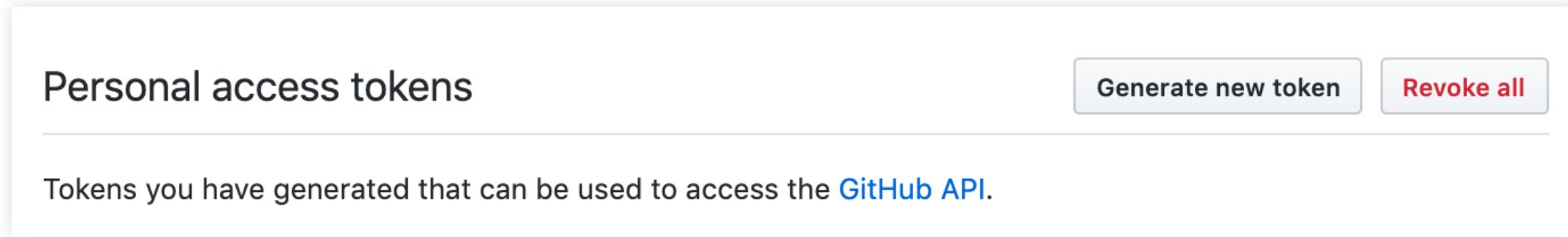


Figure 3. Token Page

# Select the public repo scope

## Note

spinnaker-instance

What's this token for?

## Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

- |   |                                      |
|---|--------------------------------------|
| <input type="checkbox"/> <b>repo</b>            | Full control of private repositories |
| <input type="checkbox"/> repo:status            | Access commit status                 |
| <input type="checkbox"/> repo_deployment        | Access deployment status             |
| <input checked="" type="checkbox"/> public_repo | Access public repositories           |
| <input type="checkbox"/> repo:invite            | Access repository invitations        |

## COPYING THE TOKEN

- You have one shot to copy the token and secure it
- Copy it, by clicking on the clipboard icon next to the hash

### Personal access tokens

[Generate new token](#) [Revoke all](#)

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ 9296031cba0e14fa0caf48f670e7d63a2ef29d9		<a href="#">Delete</a>
---	---	------------------------

# SETTING UP THE TOKEN IN SPINNAKER

Paste the token and store it in a temporary file

```
pbpaste > token_file.txt
```

Setup some temporary fields for both the location of token file, and the github account name

```
TOKEN_FILE=~/token_file.txt  
ARTIFACT_ACCOUNT_NAME=<github-username>-account
```

# INSTALLING YOUR GITHUB CONNECTION

```
hal config features edit --artifacts true  
hal config artifact github enable
```

Add your account:

```
hal config artifact github account add $ARTIFACT_ACCOUNT_NAME \  
--token-file $TOKEN_FILE
```

# DEPLOY TO SPINNAKER

```
hal deploy apply
```

# **LAB 2: DEPLOYING EKS MICROSERVICES**

# EXPRESSIONS

- Dynamically set and access variables during pipeline execution
- Any text field can use an expression
- Use them to:
  - Use arbitrary values about the state of your system in the execution of your pipelines.
  - Turn on or off particular stages or branches of your pipeline
  - Dynamically name your stack
  - Check the status of another stage
  - Perform other operations.

# WHAT ARE EXPRESSIONS?

A pipeline expression is made up of \$ followed by opening/closing brackets: \${ }

```
 ${expression here}
```

# STRING INTERPOLATION

```
 ${expressionA}-randomString-${expressionB}
```

Given "Hello" for expressionA and world for expressionB, this will result in:

```
Hello-randomString-world.
```

## WHERE CAN EXPRESSIONS BE USED?

- Pipeline expressions anywhere in the UI where you can enter free-form text
- Exception of the pipeline **Configuration** stage with exception of Expected Artifacts and "Use Default Artifact"
- If text field is unavailable, there is a feature called **Edit as JSON**

## WHEN ARE THEY EVALUATED

- Expressions are evaluated at the beginning of that stage
- Configuration needs to be evaluated *before* expression

# HELPER FUNCTIONS

- Helper functions simplify common use cases
  - Access a particular stage by name
  - Strip non-alphanumeric characters
  - Parse JSON

functions that are available

### Manual Judgment Configuration

Instructions ⓘ

Actions

Judgment Inputs

Send Notifications

Execution Options

If stage fails  halt

fn #alphanumeric  
fn #readJson  
fn #fromUrl  
fn #propertiesFromUrl  
fn #jsonFromUrl  
fn #judgment  
fn #stage  
fn #toBoolean  
fn #toFloat  
fn #toInt  
fn #toJson

# HELPER PROPERTIES

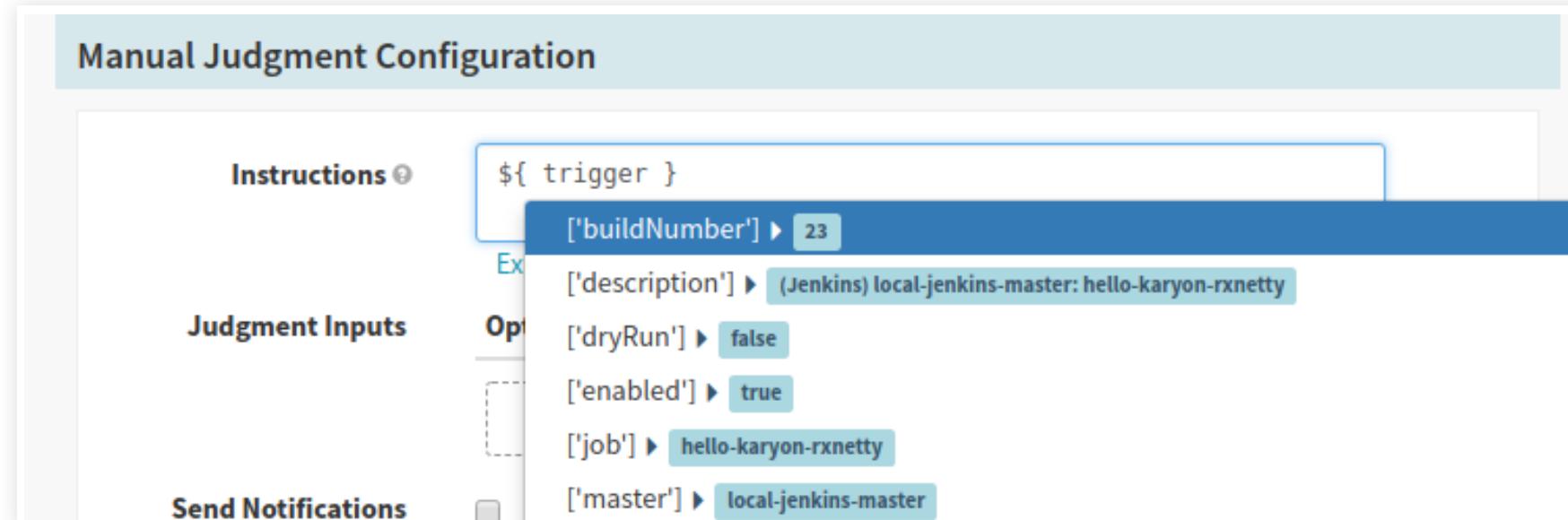
Helper properties are variables which refer to global information about the current pipeline execution.

- `execution` - refers to the current pipeline execution
- `trigger` - information about the pipeline trigger
  - e.g. Jenkins trigger and want to know which build triggered the current pipeline
  - `${trigger["buildInfo"]["number"]}`

## expression

- Type a little more of what you are looking for to narrow down your choices

! Once a helper property is added to the expression you can use any of the meta keys (Shift, Command, Alt, Control) to bring up a list of all the pipeline context that is relevant to the selected helper property.



# CONTEXT VALUES

- *Context Values* are specific to a stage.
- Includes:
  - Stage Name
  - Status
  - Start and End Time

Group by Pipeline Show 2 executions per pipeline  stage durations

MY-GCE-ACCOUNT deploy from jenkins Trigger: disabled [Configure](#) [Start Manual Execution](#)

BUILD #9  
MANUAL START [anonymous] 2 days ago [Details](#) Status: SUCCEEDED Duration: 06:49

Bake Deploy

STAGE DETAILS: BAKE Duration: 05:01

Step	Started	Duration	Status
Bake in global	2018-06-26 13:05:54 PDT	05:01	SUCCEEDED
Bake	2018-06-26 13:05:54 PDT	05:01	SUCCEEDED

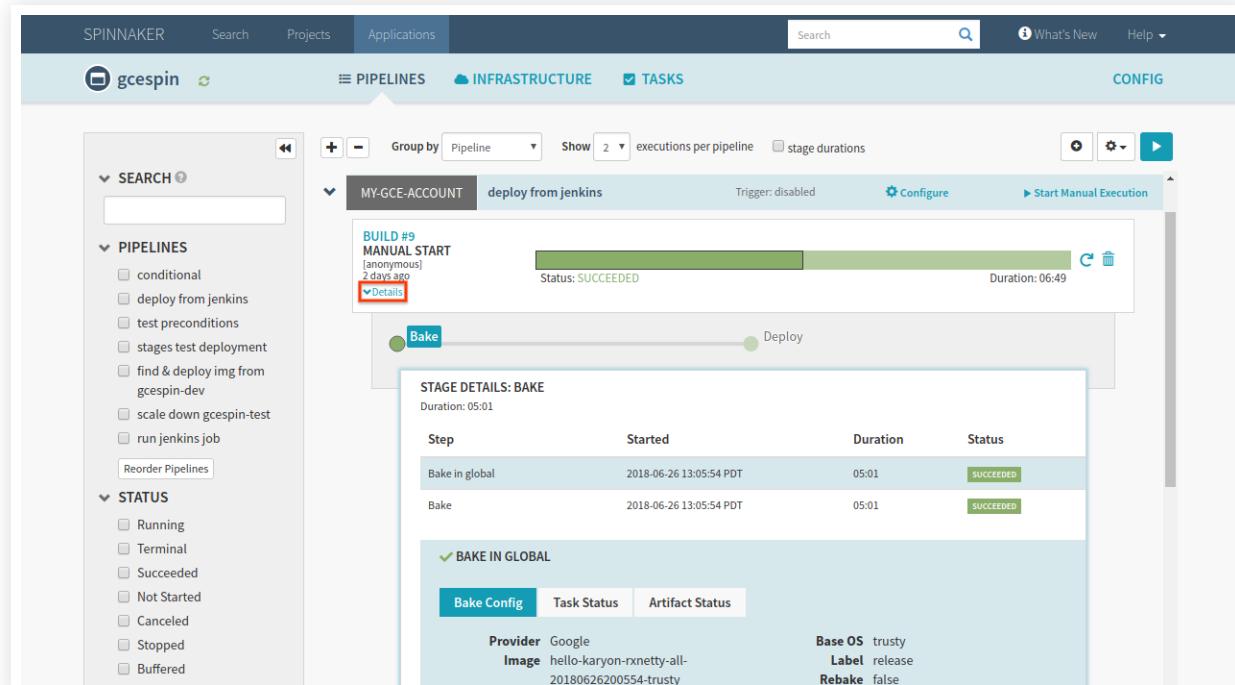
✓ BAKE IN GLOBAL

Bake Config Task Status Artifact Status

Provider	Google	Base OS	trusty
Image	hello-karyon-rxnetty-all-20180626200554-trusty	Label	release
Region	global	Rebake	false
Package	hello-karyon-rxnetty		

Image:  
hello-karyon-rxnetty-all-20180626200554-trusty

- You need a running instance of Spinnaker
- Use **Source** of any stage and identify the pipeline execution



# USING curl

You can then use `curl` to test your expression

```
PIPELINE_ID=[your_pipeline_id]
curl http://api.my.spinnaker/pipelines/$PIPELINE_ID/evaluateExpression \
      -H "Content-Type: text/plain" \
      --data '${ #stage("Deploy").status.toString() }'
```

This then returns if successful

```
{"result": "SUCCEEDED"}
```

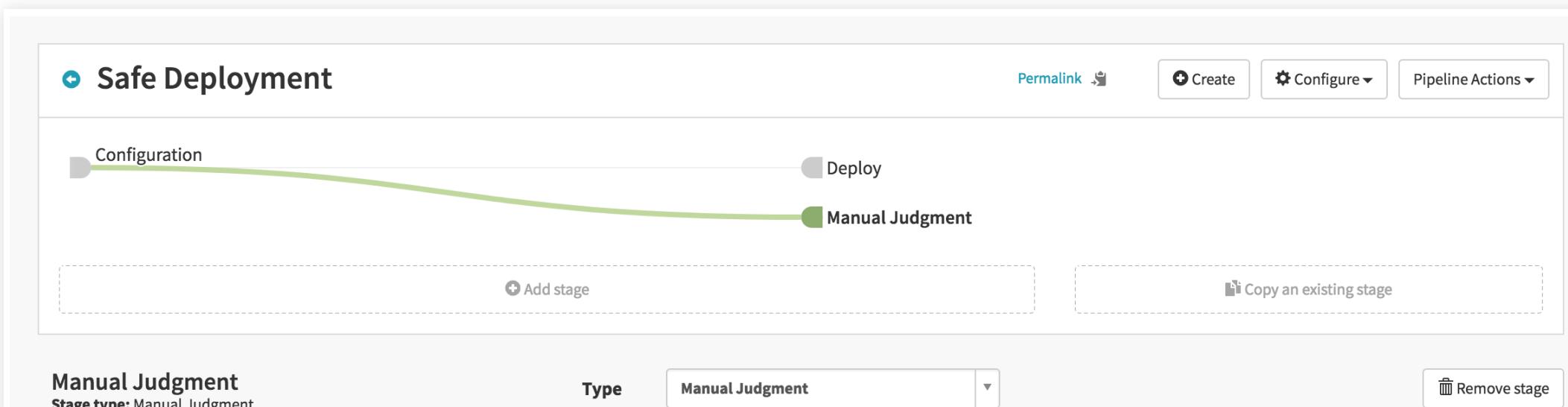
## Pipeline unsuccessful

```
{  
  "detail":  
  {  
    "{ #stage(\"Deploy\").status.toString() }":  
    [  
      {  
        "description":  
          "Failed to evaluate [expression] Expression  
          [{ #stage( #root.execution, \"Deploy\").status.toString()  
]  
          @0: No ending suffix '}' for expression starting at  
character 0:  
          { #stage( #root.execution, \"Deploy\").status.toString()  
      },  
    ]  
  }  
}
```

# MANUAL JUDGMENT

Interrupt the pipeline execution to run a manual check. This is great for teams that have a manual or out-of-band QA process

1. Click on Configuration and select Add New Stage
2. Enter “Please approve this pipeline” in the instructions



**Stage Name**

Manual Judgment

Remove stage

**Depends On**

Select...

## Manual Judgment Configuration

**Instructions**

please approve this pipeline

**Propagate Authentication** **Judgment Inputs****Option****Actions**

Add judgment input

**Send Notifications****Type****Details****Notify When****Actions**

Email

clin@netflix.com

This stage is awaiting judgment

[Edit](#)[Remove](#)

# NOTIFICATIONS WITH MANUAL JUDGMENT

1. Click on Add Notification Preference
2. In the popup dialog:
  1. Select type Email
  2. Enter your email address
  3. Select to notify when **This stage is awaiting judgment.**
  4. Click Update

ment

211362d

**Manual Judgment Instructions**  
please approve this pipeline

**Continue** **Stop**

Status: **RUNNING**

The image shows a user interface for a pipeline approval process. At the top, there is a large blue header bar with the text "WAITING FOR APPROVAL". Below this, on the left, is some dark text ("ment" and "211362d"). In the center, a white rectangular dialog box with a thin gray border is displayed. Inside the dialog, the words "Manual Judgment Instructions" are written in bold black font, followed by the text "please approve this pipeline". Below this text are two buttons: a teal-colored "Continue" button on the left and a red "Stop" button on the right. A small black icon of a person is positioned at the bottom center of the dialog. At the very bottom of the screen, there is a horizontal progress bar consisting of a yellow segment followed by a gray segment. The yellow segment is labeled "RUNNING" in blue text. A small black icon of a person is also located at the center of the progress bar.

## Manual Judgment Configuration

### Instructions ?

please approve this pipeline

### Propagate Authentication ?

### Judgment Inputs

#### Option

#### Actions

continue

[Remove](#)

rollback

[Remove](#)

 Add judgment input

### Send Notifications ?

#### Type Details

#### Notify When

#### Actions

# TRIGGERING ON CHOICE

Once applied you can create a subsequent stage that will conditionally trigger on your choice, using Expressions (SpEL)



**Conditional on Expression** 

```
judgment("Manual Judgment").equals("rollback")}
```

# **LAB 3: USING EXPRESSIONS**

## BAKING

Baking turns templates into manifests with the help of a templating engine. Helm relies on the `helm template` command to produce yaml files

# REMINDER ON HELM TEMPLATES

We can take any `yaml` resource meant for Kubernetes

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mychart-configmap
data:
  myvalue: "Hello World"
```

# TEMPLATIZE HELM TEMPLATES:

We can then templatize it with a release, called *release name*

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
```

`{{ .Release.Name }}`  will be replaced by the name of the release when the end user runs

## DEBUGGING OUR CHART

```
helm install --debug --dry-run goodly-guppy ./mychart
```

# RESULT CHART

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: goodly-guppy-configmap
data:
  myvalue: "Hello World"
```

# HELM CHARTS OFFERS A LOT

- Values
- Defaults
- Template Functions and Pipelines
- Flow Control
- Variables
- Embedded Templates

Read more: [https://helm.sh/docs/chart\\_template\\_guide/getting\\_started/](https://helm.sh/docs/chart_template_guide/getting_started/)

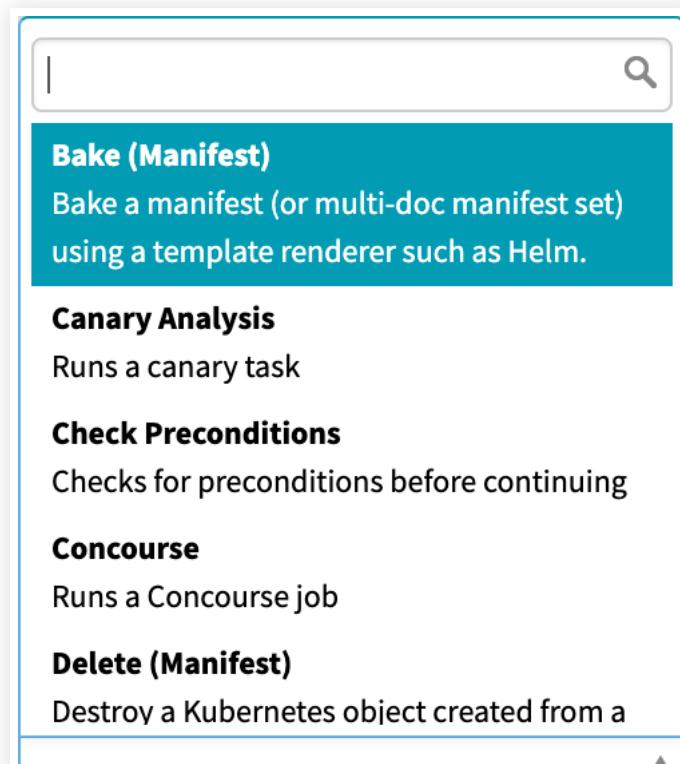
# INTENTIONS

This stage is intended to help you package and deploy applications that you own, and are actively developing and redeploying frequently. It is not intended to serve as a one-time installation method for third-party packages. If that is your goal, it's arguably better to call `helm install` once when bootstrapping your Kubernetes cluster.

## CONFIGURE ARTIFACT SUPPORT IN HAL

- Make sure that you have configured artifact support in Spinnaker first in Halyard
- All Helm charts are fetched/stored as artifacts in Spinnaker

- Create a "Bake (Manifest)" stage



### Template Renderer

Render Engine [?](#) HELM2

#### Helm Options

Name	canary-microservice
Namespace	microservices

#### Template Artifact

Expected Artifact [?](#)

#### Overrides

Expected Artifact

[+Add value artifact](#)

Overrides	Key	Value
		<a href="#">+Add override</a>

Raw Overrides [?](#)

- Enter the release name (required) - The Helm release name for this chart. This determines the name of the artifact produced by this stage
- The template artifact (required) - The Helm chart that you will be deploying, stored remotely as a `.tar.gz` archive
- The release namespace - The Kubernetes namespace to install release into. If parameter is not specified default namespace will be used
- Zero or more override artifacts (optional) - The files passed to `--values` parameter in the helm template command. Each is a remotely stored artifact representing a Helm Value File.
- Statically specified overrides - The set of static `key/value` pairs that are passed as `--set`

# HELM CHARTS AND NAMESPACE

Make sure you have `namespace` declaration in your helm chart if you are overriding the  
`namespace`

```
metadata:  
  namespace: {{ .Release.Namespace }}
```

- Spinnaker has automatically created an `embedded/base64` artifact
- The artifact is bound when the stage completes
- Represents the fully baked manifest set to be deployed downstream



## Now that the manifest set has been baked by Helm

- Configure a downstream stage (in the same pipeline or in one triggered by this pipeline)
- Deploy the artifact produced by the “Bake (Manifest)” stage

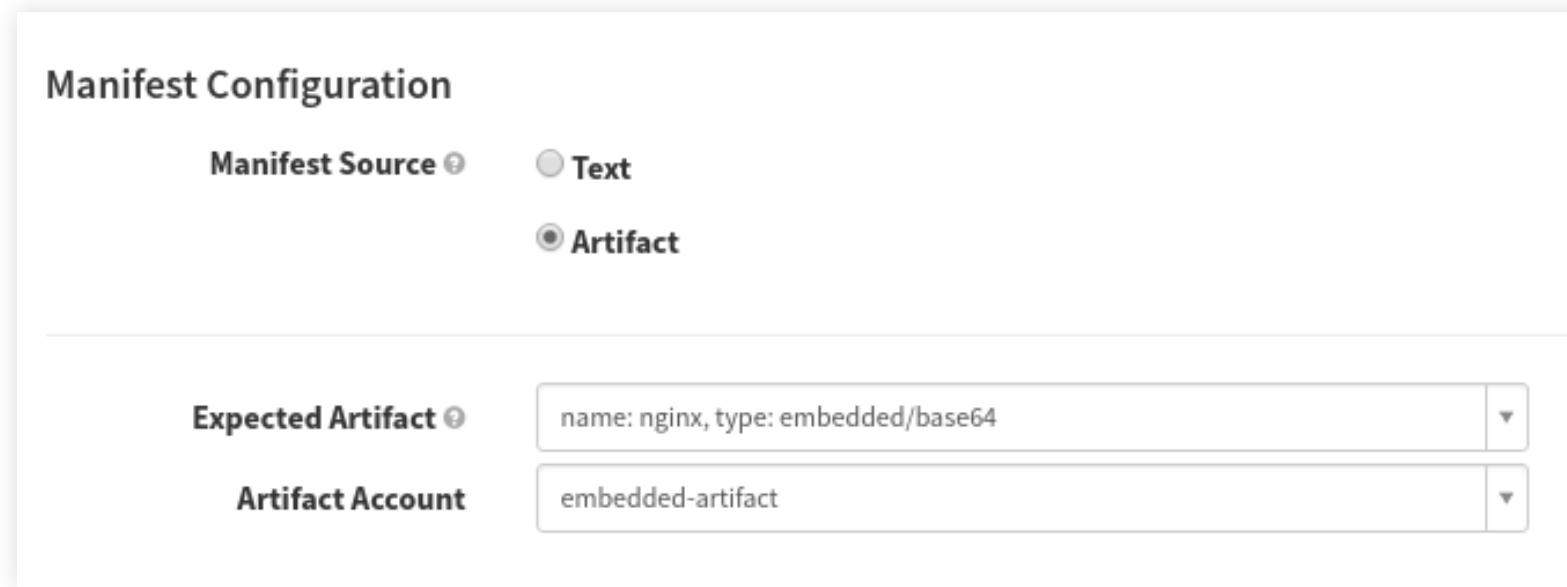
Manifest Configuration

Manifest Source  Text  Artifact

---

Expected Artifact  name: nginx, type: embedded/base64

Artifact Account  embedded-artifact

A screenshot of a user interface titled "Manifest Configuration". It shows two radio button options for "Manifest Source": "Text" (unchecked) and "Artifact" (checked). Below this is a section for "Expected Artifact" with a dropdown menu containing the text "name: nginx, type: embedded/base64" and a "▼" button. Another section below it is for "Artifact Account" with a dropdown menu containing the text "embedded-artifact" and a "▼" button.

# **LAB 4: DEPLOYING HELM MICROSERVICES**

# ROLLBACKS

## AUTOMATIC ROLLBACKS

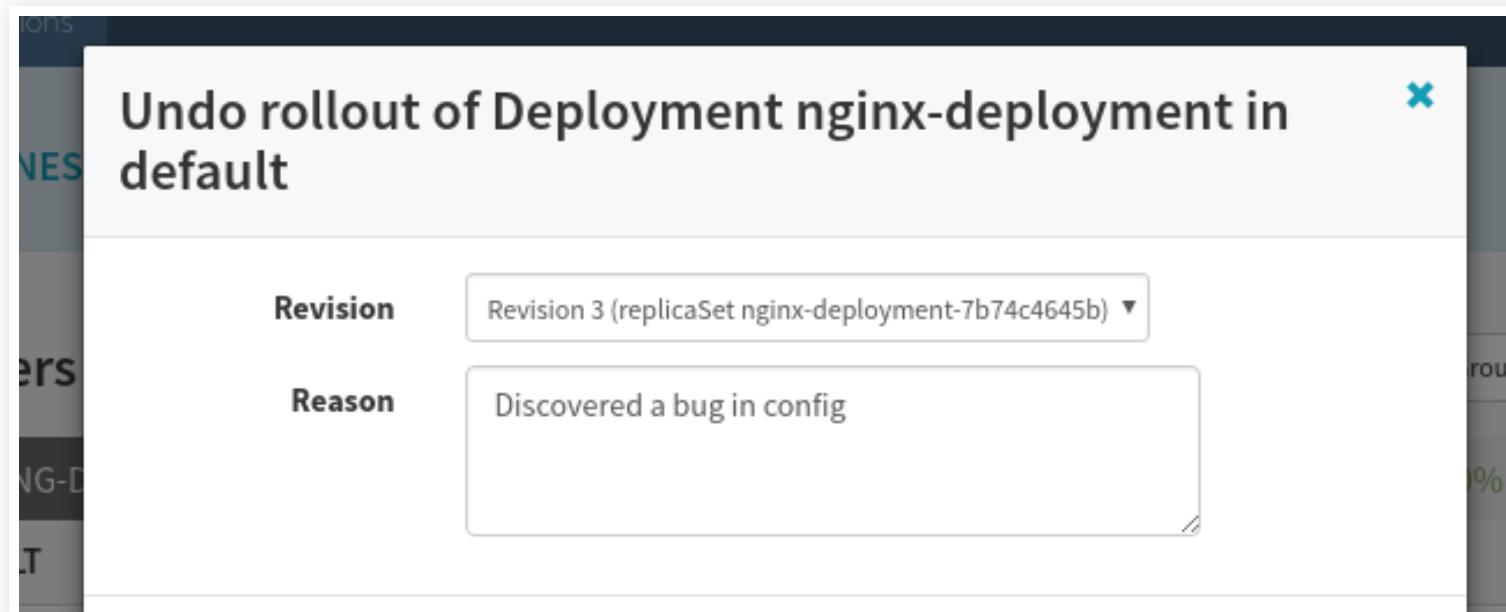
- Having manifests is optimal, IaC
- Rollbacks though are necessary when things go wrong

## Application

- Spinnaker exposes “Undo Rollout” functionality in two places:
  - Clusters tab
  - Pipeline stage

## “Clusters” tab

- You can select “Undo Rollout” from the “Actions” dropdown
- This provides an opportunity to select a healthier alternative



- These stages are best configured to run when other stages or branches fail, indicating that something has gone wrong in your rollout.

- Number of Revisions Back:

- Counts how many revisions the current active revision should be rolled back by. If you have the following state:

```
nginx-deployment-2d8178b77 (Revision 5) # active
nginx-deployment-7bdd110f7 (Revision 4)
nginx-deployment-0b13cc8c1 (Revision 1)
```

And roll back by “1” revision, (Revision 4) will be active again. Roll back by “2” revisions and (Revision 1) will be active again.

# PARAMETERIZED ROLLBACKS

- Parameterize the target resource to roll back
- Reference anything via pipeline expressions
- Point to something specified using pipeline parameters, for example:
  - Upstream deploy stage
  - Another stage's outputs
  - Using pipeline expressions.

# VERSIONING ARTIFACTS

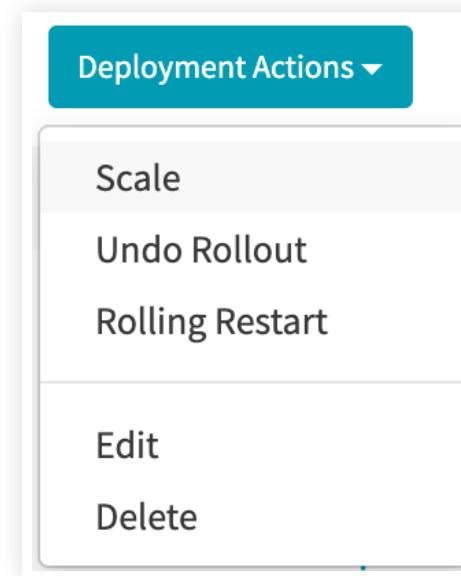
- Rolling Back will not work unless artifacts are versioned, this includes:
  - Docker images
  - ConfigMap
  - Kubernetes Secrets

Source: <https://blog.questionable.services/article/kubernetes-deployments-configmap-change/>

# SCALING

# SCALING RESOURCES

From the **Clusters** Page and selecting the Deployment you can scale the Replica Sets supports



# SELECT THE REPLICAS

You can then select the number of replicas you wish to either increase or decrease

Scale Deployment large-breed-microservice in microservices ×

<b>Replicas</b>	3
<b>Reason</b>	(Optional) anything that might be helpful to explain the reason for this change; HTML is okay

Cancel Submit

- ! If you want the number of replica sets permanent, be sure to update your yaml file, changes will not be persisted

## ROLLOUT STRATEGIES

- Deploys your containers as either:
  - Dark Rollouts
  - Highlander
  - Red/Black (also known as Blue/Green)
- Is only valid for Kubernetes `ReplicaSet`
- Verifies health checks
- Disables old server groups
- Enabling new server groups

- Configuration options allow you to associate a workload with one or more services
- Decide whether the workload should receive traffic, and determine how Spinnaker should handle any previous versions of the workload in the same cluster and namespace

Deploy (Manifest) Configuration

Rollout Strategy Options

<b>Enable</b> ⓘ	<input checked="" type="checkbox"/> Spinnaker manages traffic based on your selected strategy
<b>Service(s) Namespace</b>	Select...
<b>Service(s)</b>	Select...
<b>Traffic</b>	<input type="checkbox"/> Send client requests to new pods
<b>Strategy</b> ⓘ	None

- Service namespace - Select the namespace containing the service(s) you would like to associate with the workload.
- Service(s) - Select one or more services you would like to associate with the workload. Spinnaker will add a `traffic.spinnaker.io/load-balancers` annotation listing the selected services as described here.
- Traffic - Check this box if you would like the workload to begin receiving traffic from the selected services as soon as it is ready. If you do not check this box, you can add a subsequent Enable (Manifest) stage to begin sending traffic to the workload.
- Strategy - Select a strategy if you would like Spinnaker to handle previous versions of the workload currently deployed to the same cluster and namespace. Select None if you do not want Spinnaker to take any action regarding existing workloads.

# STRATEGIES

- Use a dark rollout to deploy a new version of your application alongside the existing version(s)
- Does not immediately route any traffic to the new version.
- Add subsequent Enable (Manifest) and Disable (Manifest) stage to begin sending traffic to the new version and stop sending traffic to the old version(s).

Rollout Strategy Options

<b>Enable</b> ⓘ	<input checked="" type="checkbox"/> Spinnaker manages traffic based on your selected strategy
<b>Service(s) Namespace</b>	default
<b>Service(s)</b>	x maggie-k8s-demo
<b>Traffic</b>	<input type="checkbox"/> Send client requests to new pods

*"There can be only one" – Highlander (1986)*

- Deploy a new version of your application alongside the existing version(s)
- Send client traffic to the new version
- Then disable and destroy existing versions in the cluster

**Rollout Strategy Options**

---

<b>Enable</b> <small>?</small>	<input checked="" type="checkbox"/> Spinnaker manages traffic based on your selected strategy
<b>Service(s) Namespace</b>	default
<b>Service(s)</b>	<input type="text" value="maggie-k8s-demo"/> <small>x</small>
<b>Traffic</b>	<input type="checkbox"/> Send client requests to new pods

version(s)

- Send client traffic to the new version, and then disable existing versions in the cluster
- Optionally, add subsequent Destroy (Manifest) stages to clean up any unwanted workloads in the cluster
- Alternately, easily roll back to a previous version by:
  - Configuring an **Enable (Manifest)** stage
  - Or using an ad-hoc Enable operation from the Clusters tab.

#### Rollout Strategy Options

**Enable** ?  Spinnaker manages traffic based on your selected strategy

# RESTRICT EXECUTION

can happen.

- By using execution windows, you can ensure that deployments don't interfere with times where your service is at peak demand.
  - You can also use execution windows to make sure that there is always someone in the office ready to manually intervene or rollback your pipeline.

## Execution Options

**Restrict execution to specific time windows**

**Days of the Week**(No days selected implies execution on any day if triggered)

Sun	Mon	Tue	Wed	Thu	Fri	Sat
-----	-----	-----	-----	-----	-----	-----

All    None    Weekdays    Weekend

**Time of Day**



# CANARY DEPLOYMENT

# WHAT IS IT?

- Canary is a deployment process in which a change is partially rolled out then evaluated against the current deployment (baseline)
- Ensure that the new deployment is operating at least as well as the old.
- This evaluation is done using key metrics that are chosen when the canary is configured.
- Microservices would need to expose some of their metrics
- Canaries are usually run against deployments containing changes to code, but they can also be used for operational changes, including changes to configuration.

- Have metrics to evaluate
  - Provide metrics in your microservice application
  - Either deploy metrics or have it scraped
  - Support is built in for:
    - StackDriver
    - Datadog
    - Prometheus
    - SignalFx

# PROCESS

- Enable Canary
- Create one or more canary configurations
- Add one or more canary stages

## ENABLE CANARY

- Use Halyard
- Setup *one* Metrics Service
- Setup *one* a Storage Service

# SETTING UP CANARY ANALYSIS

```
hal config canary enable
```

## SETTING UP THE SCOPE

- Each configuration is visible to all canary pipeline stages
- It can be configured so that it is only visible to the application where it was created

```
hal config canary edit --show-all-configs-enabled false
```

# SETTING UP THE CANARY JUDGE

- The current default judge is `NetflixACAJudge-v1.0`
- A judge accesses the quality of your deployment against the baseline
- Each is compared to determine any degradation
- Other judges are available and are pluggable

```
hal config canary edit --default-judge JUDGE
```

# IDENTIFY YOUR METRICS PROVIDER

```
hal config canary edit --default-metrics-store STORE
```

STORE **can** be:

- atlas
- datadog
- stackdriver
- prometheus
- newrelic

## PROVIDE A DEFAULT METRICS ACCOUNT

- Add the account name to use for your metrics provider
- Default can be overridden

```
hal config canary edit --default-metrics-account ACCOUNT
```

# PROVIDE THE DEFAULT STORAGE ACCOUNT

- Add the account name to use for your storage provider
- Default can be overridden

```
hal config canary edit --default-storage-account ACCOUNT
```

# SETTING UP PROMETHEUS

```
hal config canary prometheus enable
```

# MANAGE OR VIEW PROMETHEUS ACCOUNT FOR CANARY

Add a Halyard/Spinnaker account to Prometheus

```
hal config canary prometheus account add ACCOUNT --base-url
```

Edit a Halyard/Spinnaker account to Prometheus

```
hal config canary prometheus account edit ACCOUNT --base-url
```

Operations for `delete` and `list` are also available

# CREATE A CANARY CONFIGURATION

- Canary configuration is done per *application*
- Each Application will have one or more configurations
- Stages are defined separately

# WHAT YOU WILL BE ADDING INTO A CANARY CONFIGURATION

You will be adding the following to your canary configuration:

- A name by which a canary stage can choose this configuration
- The specific metrics to evaluate, and a logical grouping of those metrics
- Default scoring thresholds (which can be overridden in the canary stage)
- Optionally, one or more filter templates

- Do this separately for all applications that will use automated canary analysis

## Features

If you don't need or want certain features for your application, you can disable them here. Disabling a feature only changes the display in Spinnaker - it won't delete any actual data.

- Pipelines**  
Orchestrated deployment management
- Clusters**  
Collections of server groups or jobs
- Load Balancers**  
Traffic distribution management between servers
- Security Groups**  
Network traffic access management
- Canary**  
Canary analysis configuration and reporting



## CREATE YOUR CANARY CONFIGURATION

- Configurations you create within an application are available to all pipelines in that application
- Unless it is setup that all configurations are available to all applications

# STEPS FOR CANARY CONFIGURATION SUPPORT

## Hover over the Delivery Tab, select Canary Configs

The screenshot shows the Spinnaker application interface for managing an application named "myapp". The top navigation bar includes links for SPINNAKER, Search, Projects, Applications, and a user icon. Below the navigation is a header with the application name "myapp" and icons for edit and refresh. The main content area has tabs for DELIVERY, INFRASTRUCTURE, and TASKS. The DELIVERY tab is active, with its sub-menu expanded to show PIPELINES, CANARY CONFIGS, and CANARY REPORTS. The CANARY CONFIGS item is currently selected. On the left, there's a sidebar with links for APPLICATION ATTRIBUTES, NOTIFICATIONS, FEATURES, LINKS, CACHE MANAGEMENT, TRAFFIC GUARDS, and DELETE APPLICATION. The right side displays application attributes, including Owner (duftler@google.com), Description (my-gce-account), Account(s) (my-gce-account), Cloud Provider(s) (aws, gce, appengine), Instance health (This application considers only cloud provider health when executing tasks.), Instance Port (79), and User Data Format (This application requires legacy user data format.). At the bottom of this section is a link to "Edit Application Attributes".

Attribute	Description
Owner	duftler@google.com
Description	my-gce-account
Account(s)	my-gce-account
Cloud Provider(s)	aws, gce, appengine
Instance health	This application considers only cloud provider health when executing tasks.
Instance Port	79
User Data Format	This application requires legacy user data format.

[Edit Application Attributes](#)

# SELECTING YOUR CONFIGURATION

- Select **Add Configuration**
- Provide a name and a description. The name is displayed when you show the canary stage on pipeline
- Select your telemetry provider

## CREATE METRICS GROUP

1. Create groups to determine weights
2. Click Group to create each group you'll use. Select the group and click the edit icon
3. An example, would be "cpu" group to add a set of cpu metrics

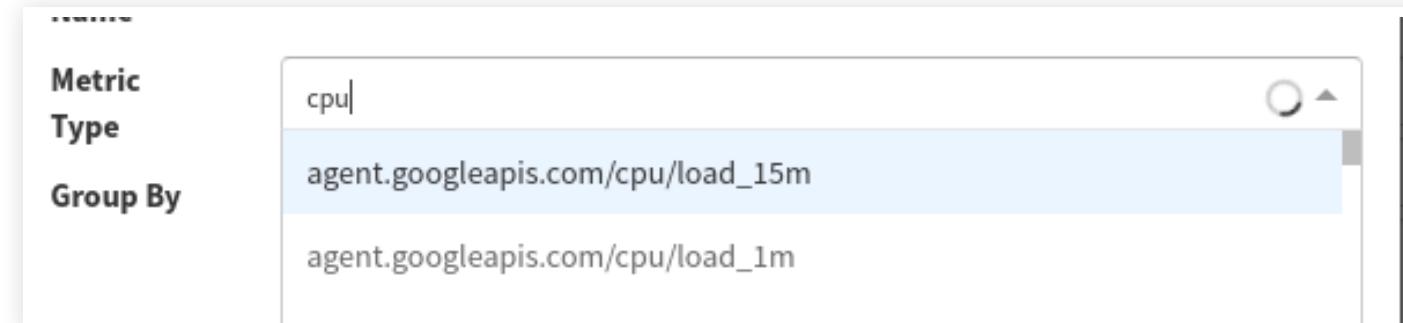
# CREATE METRICS

1. In the **Metrics** section, add **Metric**
2. Select the group to add this metric
3. Give the metric a name
4. Specify whether this metric fails when the value deviates too high or too low, or both compared to the baseline
5. Optionally, choose a filter template

```
resource.type = "gce_instance" AND  
resource.labels.zone = starts_with("${zone}")
```

- Optionally, if your provider supports aggregation, you can click **Group by** and enter the metric metadata
- For example, when you create a metric you can group its time series by resource or metric label. You can group a time series by zone, for example (`resource.zone`)

**i** When you create a canary configuration, you create metric groups, and scoring thresholds and weights are applied to groups (rather than to specific metrics). But the grouping described in this step is for aggregating metrics before they're returned to Kayenta.



If you are using *StackDriver* or *Prometheus* you can add filter templates then assign it to a metric

1. Click **Add Template**
2. Provide a Name - This is the name by which you can select it when configuring the specific metric.
3. In the Template field, enter an expression using the *FreeMarker* template language [[https://freemarker.apache.org/docs/dgui\\_quickstart\\_template.html](https://freemarker.apache.org/docs/dgui_quickstart_template.html)]
4. The expression is expanded using the variable bindings specified via the Extended Params in any canary stage that uses this configuration.
5. These variable bindings are also implicitly available: project, resourceType, scope,

## EDITING A CONFIGURATION

- In **Delivery** tab select **Canary Configs**
- Select the configuration you would like to edit

# ADDING A CANARY STAGE

## ABOUT CANARY STAGES

Canary analysis can be performed over data points collected beginning from the moment of execution and into the future, or it can be performed over a specified time interval.

## REAL-TIME ANALYSIS

- A real-time analysis means that the canary analysis is performed over a time interval beginning at the moment of execution.
- The analysis happens for a specified time period, beginning when the stage executes (or after a specified Delay).
- For Real Time, also specify the number of hours to run (Lifetime).

# RETROSPECTIVE ANALYSIS

- In a retrospective analysis the canary analysis is performed over an explicitly specified time interval (likely in the past).
- Analysis occurs over some specified period. Typically, this is done for a time period in the past, against a baseline deployment and a canary deployment which have already been running before this canary analysis stage starts.
- Note that this analysis might analyze data for resources which no longer exist, for which there are still published time series.

# METRIC SCOPE

Metric scope defines:

- Where, when, and on what the canary analysis occurs.
- Specific baseline and canary server groups
- Start and end times and interval
- Cloud resource on which the baseline and canary are running

# DEFINING THE CANARY STAGE

Canary Config

---

Analysis Type ⓘ Retrospective

Analysis Config

---

Config Name MySampleStackdriverCanaryConfig

Interval ⓘ 180 minutes

Analysis Type ⓘ Growing

# DEFINING THE CANARY STAGE DETAILS

- In your pipeline click on **Add Stage**
- For type, select **Canary**
- Give it the stage name and depends on field
- In the analysis type select **Real Time** or **Retrospective**
- Choose a configuration name for the configuration you created before adding this stage

# DEFINING THE CANARY STAGE DETAILS CONTINUED

- Set a delay: How many minutes to wait until warmed up
- Set an interval: This is how frequently (in minutes) to capture and score the metrics.
- Lookback Type: Growing or Slowing
  - In a growing analysis, a judgment is taken every [interval] minutes, but each judgment goes all the way back to the beginning of the Lifetime.
  - A sliding Lookback also makes a judgment every [interval], but each judgment only looks at the data from the most recent lookback duration. It would not be unusual for the Interval and the look-back duration to be the same, but they don't have to be.



**Baseline** myapp-v059

**Baseline Region** us-central1

**Canary** myapp-v059

**Canary Region** us-central1

**Step** 60 seconds

**Resource Type** gce\_instance ▾

**Extended Params** Key Value

[ ] Add Field

- Adjust Scoring Thresholds is needed - The thresholds are pre-populated based on those configured in the main canary config, but you can override them here.

- Baseline - The server group to treat as the “control” in the canary analysis—that is, the deployment against which to compare the canary deployment.
- Baseline Region - The region in which the baseline server group is deployed.
- Canary - The server group to treat as the experiment in the analysis.
- Canary Region - The region in which that canary server group is deployed.
- Step - The interval, in seconds, for the metric time series.
- Start Time and End Time (for retrospective) For a retrospective analysis, the specific time span over which to conduct the analysis.
- Extended Params - Add any additional parameters, which are specific to the metric sources and which can be used to refine the scope of the analysis. These parameters can

Judgment goes through two phases:

- Data collection
  - Collect data from both baseline and canary deployments
  - Stored in a timeseries database
  - Include a set of tags or annotations that identify deployment
- Judgment
  - Compare the metrics from the data collection
  - Renders a pass or fail

# JUDGMENT PHASES

# DATA VALIDATION

- Ensures that there is data
- If no data is found in either baseline or canary, it moves onto the next metric
- Judge does not fail if there is no data, since some data it's ok to have no data

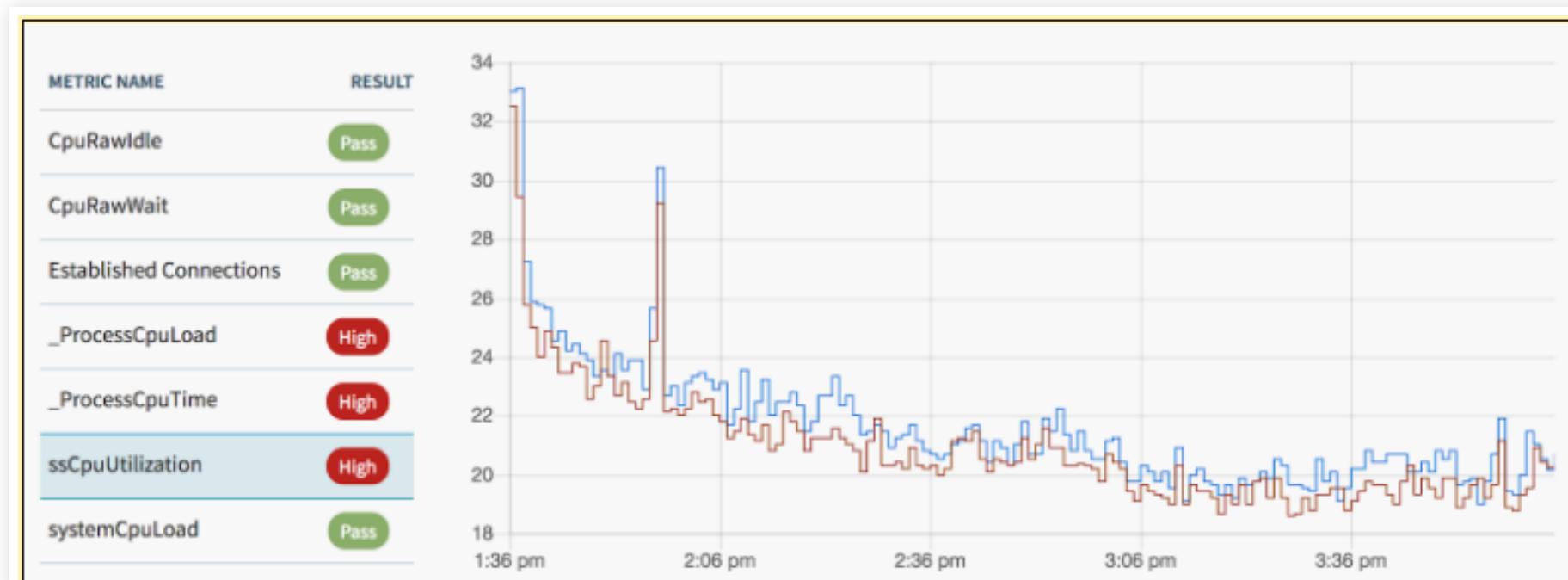
# DATA CLEANING

- Handling Missing Values
- Impute values for missing values
- Optionally, outliers can be removed

## METRIC COMPARISON

- This is the step that compares the canary and baseline data for each included metric.
- Classification indicating there is a significant difference between baseline and canary
- Each metric is either *Pass*, *High*, or *Low*
- Uses nonparametric statistical test to check for a significant difference between the canary and baseline metrics

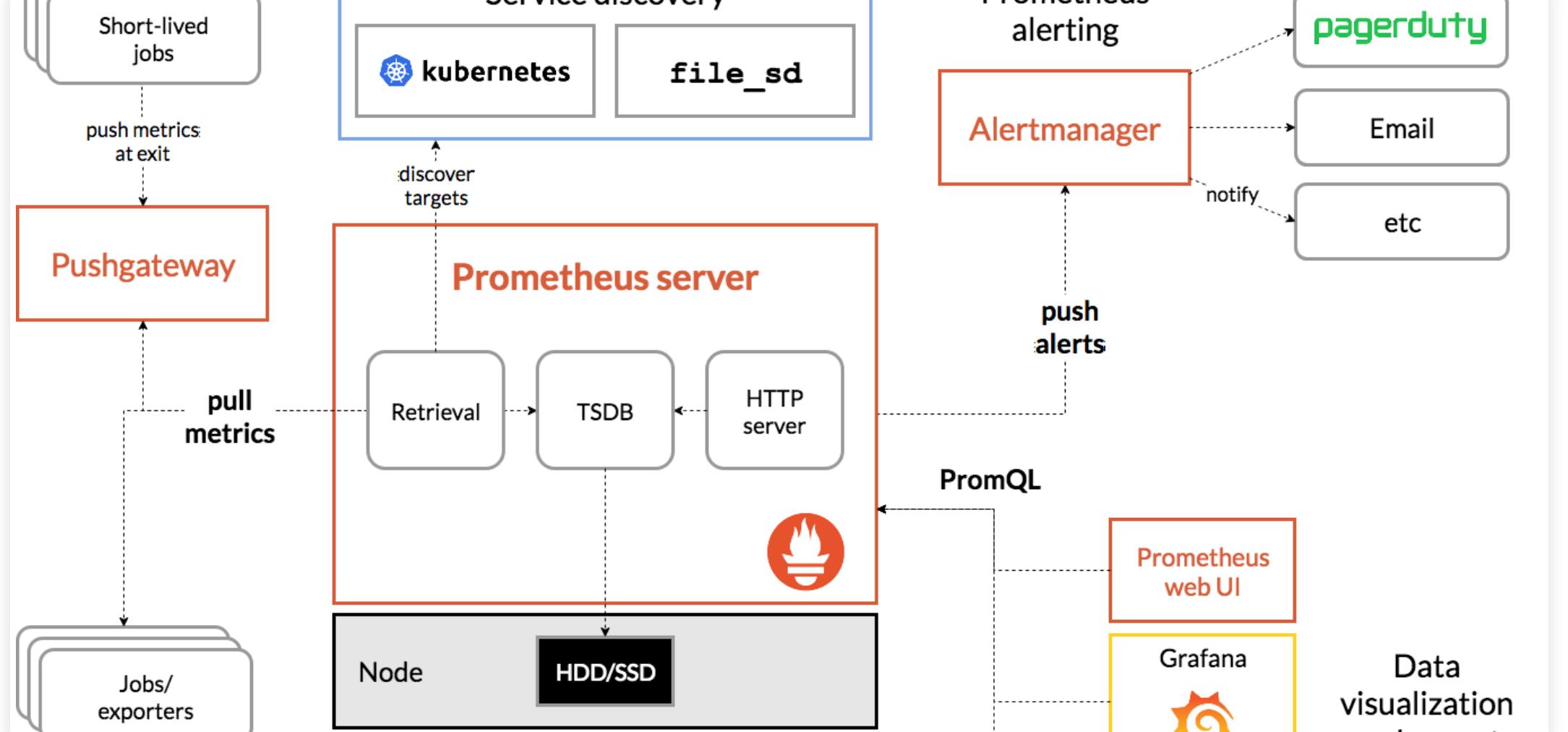
- If 9 out of 10 is "Pass" it is classified as pass if specified in the root configuration
- Default judge is biased for simplicity- easy to interpret and understand



# PROMETHEUS

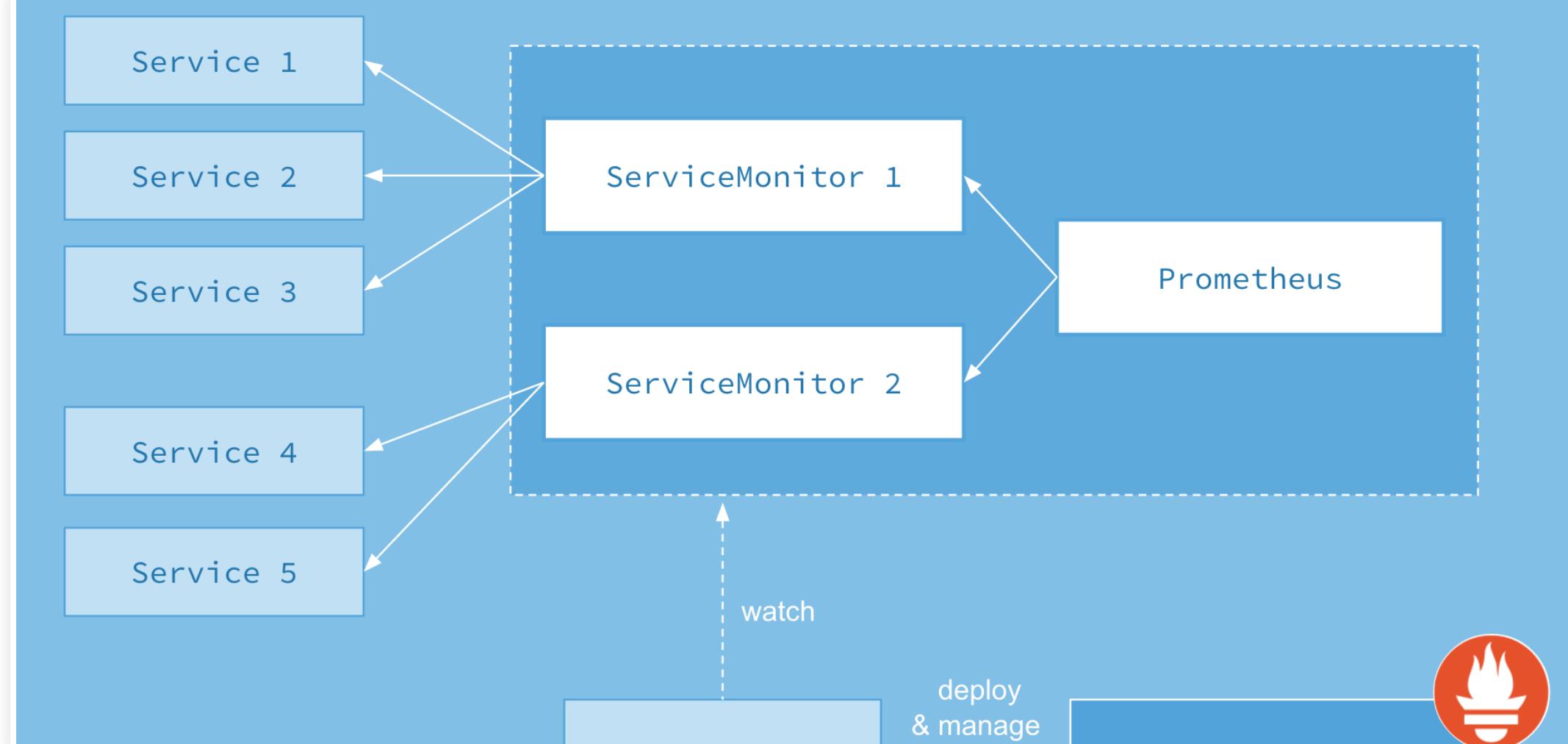
# PROMETHEUS OVERVIEW

- Prometheus is just one of the available metrics packages that analyzes data from your microservice
- There are various metrics packages to include in your microservice in a variety of languages
- The metrics are "scraped" by Prometheus and aggregated into reports



# PROMETHEUS SETUP IN KUBERNETES

- There are various utilities to get Prometheus up and running, and documentation is terrible
- One of the easier ones, despite horrible documentation, is the Prometheus Operator on Helm(<https://coreos.com/operators/prometheus/docs/latest/user-guides/getting-started.html>)



# SETUP FOR PROMETHEUS

```
helm install <name> stable/prometheus-operator
```



⚠ Ensure that you are in the right context. For canary testing you would need to be in `staging`

- `namespaceSelector` will look for metrics in other namespaces
- This `ServiceMonitor` needs to be deployed in the default namespace

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    release: lasertag
  name: teama-microservice-service-monitor
spec:
  endpoints:
    - interval: 30s
      path: /metrics
```

## Once established, deploy a NodePort or LoadBalancer

```
apiVersion: v1
kind: Service
metadata:
  name: prometheus
spec:
  type: (NodePort | LoadBalancer)
  ports:
  - name: web
    nodePort: 30900
    port: 9090
    protocol: TCP
    targetPort: web
  selector:
    app: prometheus
```

# **LAB 5: ADVANCED FEATURES**

# TERRAFORM DEPLOYMENTS

# INFRASTRUCTURE AS CODE

- Infrastructure as code is the process of managing and provisioning computer data centers through machine-readable definition files
- As opposed to physical and manual hardware configuration or interactive configuration tools.
- Typically stored in version control so that configuration can be managed and reviewed
- Provides Speed, Consistency, and Accountability

# ABOUT TERRAFORM

- Terraform is a software that can:
  - Provision cloud resources
  - Manage Infrastructure and Services
- Created by Hashicorp (<https://www.terraform.io/>)

## HOW IT WORKS

- Configuration based Infrastructure as Code
- Describe the components as to the desired state
- Uses Idempotency to update that state

## KEY FEATURES

- **Infrastructure as Code** - Using files to describe the infrastructure you wish to create
- **Execution Plans** - Shows what Terraform will do, when applied
- **Resource Graph** - Terraform builds a graph of all resources and performs in parallel
- **Change Automation** - Since it is code, changes can be applied with little human interaction

- Terraform can provision varying resources in multicloud platforms all in one infrastructure as code resource
- View the providers at <https://registry.terraform.io/browse/providers>
  - AWS
  - GCP
  - Digital Ocean
  - Kubernetes
  - Alibaba

# SETTING UP TERRAFORM FOR AWS ACCESS

Keep in mind that you can run on different cloud providers

```
$ export AWS_ACCESS_KEY_ID=(your access key id)  
$ export AWS_SECRET_ACCESS_KEY=(your secret access key)
```



Terraform can also use `$HOME/.aws/credentials` as well IAM profile with EC2 Instance

- `profile` refers to the AWS credentials file
- `provider` is a plugin that translates API interactions
- `resource` are resources that you can create, such as servers, databases, and load balancers

## Creating an AWS Resource and an AWS Instance

```
provider "aws" {  
    profile      = "default"  
    region       = "us-east-1"  
}  
  
resource "aws_instance" "example" {  
    ami           = "ami-2757f631"  
    instance_type = "t2.micro"
```

- Enter into the folder where your Terraform files (*.tf*) are located
- Run `terraform init`
- What it does:
  - Scans all *tffiles*
  - Figure out which providers you're requiring
  - Downloads the code for the providers
- It is safe to run multiple times

- It is similar to *diff* commands in git or bash
  - Anything with a plus (+) indicates an item will be created
  - Anything with a minus (-) indicates an item will be removed
  - Anything with a tilde (~) indicates an item will be modified

```
$ terraform plan
```

Can yield the response similar to the following

# APPLYING TERRAFORM

- Scans the current directory for the configuration and applies the changes appropriately.
- Terraform keeps track of all the resources it already created so it knows your infrastructure already exists with subsequent runs

```
$ terraform apply
```

# TERRAFORM IS MEANT FOR VERSION CONTROL

- The Terraform Configuration Files you create should be stored in git
- What you shouldn't check in is the scratch folder or state files
  - .terraform **folder**
  - \*.**tfstate**
  - \*.**tfstate.backup**

```
$ tree .terraform/
.terraform/
    └── modules
        ├── b3da51524bda220ab3e4db8209644422
        │   └── main.tf
        ├── b7aaad981fe624c7ca40237c2a778f8c
        │   └── main.tf
        └── modules.json
    └── plugins
        └── darwin_amd64
            └── lock.json
```

# TERRAFORM STATE FILES

- As you run Terraform it records information about the infrastructure it created in a *Terraform state file*.
- By default, when you run Terraform in the folder `/foo/bar`, Terraform creates the file `/foo/bar/terraform.tfstate`.
- State files contains a custom JSON format that records a mapping from the Terraform resources in your configuration files to the representation of those resources in the real world.

# WHERE TO STORE THE STATE FILE?

To work in a team environment:

- Each of your team members needs access to the same Terraform state files.
- That means you need to store those files in a shared location.

# LOCKING OF THE STATE FILES

- If two team members are running Terraform at the same time you can run into race conditions
- Multiple Terraform processes make concurrent updates to the state files
- This leads to:
  - Conflicts
  - Data loss
  - Corruption

- Create an S3 bucket via Terraform
- Create a Dynamo DB resource to manage locking

In your main project folder:

- Setup a `terraform.backend` entry where you will list:
  - `bucket` - Name of the S3 bucket
  - `key` - File Path where the state is be written
  - `region` - AWS Region

# SHARED STATE EXAMPLE IN TERRAFORM

```
terraform {  
  backend "s3" {  
    # Replace this with your bucket name!  
    bucket      = "my-team-application.tfstate"  
    key         = "global/s3/terraform.tfstate"  
    region      = "us-west-2"  
  
    # Replace this with your DynamoDB table name!  
    dynamodb_table = "my-team-application-locks"  
    encrypt       = true  
  }  
}
```

## CHICKEN AND EGG IN TERRAFORM

- First run the Terraform to set up S3 bucket and DynamoDB
- Then run your application with the backend

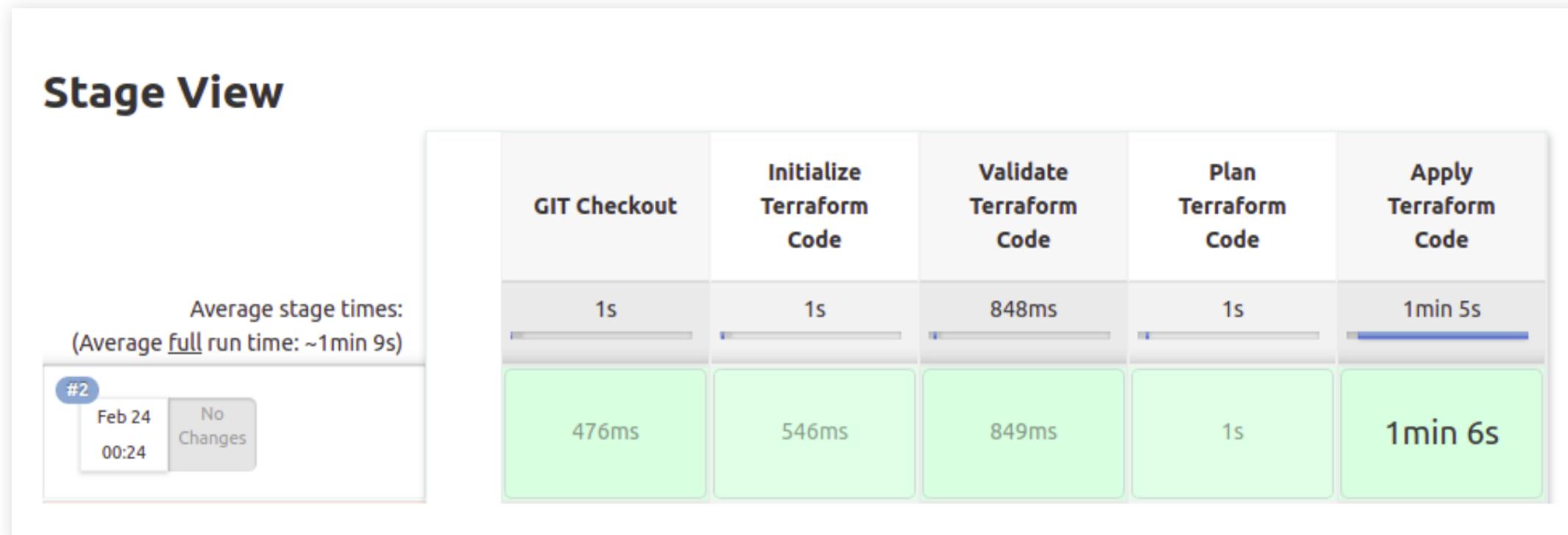
# TERRAFORM FOLDER STRUCTURE

Name	Last commit	Last update
..		
❖ .gitignore	Created iam, alb,acm, route53 modules in ...	2 days ago
❖ backend.tf	Created iam, alb,acm, route53 modules in ...	2 days ago
❖ data.tf	Created iam, alb,acm, route53 modules in ...	2 days ago
❖ main.tf	Created iam, alb,acm, route53 modules in ...	2 days ago
❖ output.tf	Created iam, alb,acm, route53 modules in ...	2 days ago
❖ provider.tf	Created iam, alb,acm, route53 modules in ...	2 days ago
❖ <u>variables.tf</u>	Created iam, alb,acm, route53 modules in ...	2 days ago

# WHERE CAN WE RUN TERRAFORM?

- Client (your) machines
- Cloud Instances
- CI/CD Frameworks

- There are many choices for you to make where you decide to run Terraform
- Some choices are better than others in different situations



- You need the output local to your pipelines so as to perform other operations
- You are wanting to setup infrastructure as a Kubernetes Job
- You don't want Terraform run on client machines and require more control (CI can also be used for this purpose)
- View everything in one single place. More Manageable, More Visibility.

### Arguments Against:

- Makes more sense to create infrastructure in CI rather than CD for locality purposes
- Operators prefer to run Terraform on local machines

# TERRAFORM IN SPINNAKER SOLUTIONS

- Kubernetes Job
- Armory.io
- OpsMx Terraspin

# KUBERNETES JOB

- Single purpose work
- A `Job` creates one or more Pods
- Will continue to retry execution of the Pods until a specified number of them successfully terminate.
- The Job object will start a new Pod if the first Pod fails or is deleted (for example due to a node hardware failure or a node reboot).
- You can also use a Job to run multiple Pods in parallel.

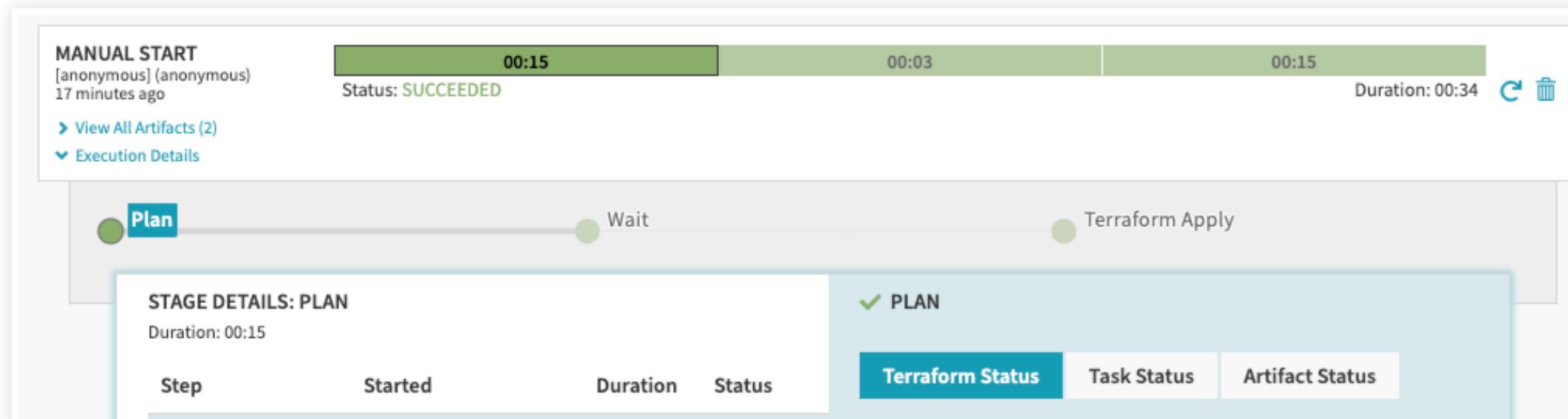
# EXAMPLE KUBERNETES JOB

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
        - name: pi
          image: perl
          command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
  backoffLimit: 4
```

# RUNNING TERRAFORM AS A KUBERNETES JOB

- Since Kubernetes can run anything, that would include a Terraform job
- What is required?
  - Creating a container that is derived from a Terraform image  
<https://hub.docker.com/r/hashicorp/terraform>

- This performs `terraform init` and `terraform apply` as separate stages
- Includes state management
- Part of their proprietary platform, not part of the OSS Platform



# OPSMX TERRASPIN

- OpsMX, another Spinnaker company also created some custom stages for Spinnaker
- Installation includes a Kubernetes ConfigMap and setting up microservice files through halyard
- Includes State Management to S3
- Provide three stages:
  1. TSPlanRest
  2. TSApplyRest
  3. TSDestroyRest

**TerraformAzurek8DemoRest**

The screenshot shows a pipeline configuration interface for a project named "TerraformAzurek8DemoRest". The pipeline consists of several stages connected sequentially: Configuration, TSPlanRest, Plan-Validation, Deploy (Manifest), User-Validation, and TSDestroyRest. A new stage is currently being added at the end of the pipeline. A search bar at the top right is active, displaying the prefix "res". A dropdown menu lists several stage types, with "TSPlanRest" selected and highlighted in blue. Other visible stage types include "TSApplyRest", "TSDestroyRest", "Find Artifacts From Resource (Manifest)", and "Entity Tags". The "Find Artifacts From Resource (Manifest)" stage is described as finding artifacts from a Kubernetes resource. The "Entity Tags" stage is listed under "Type". Below the search bar, there are buttons for "Permalink", "Pipeline Actions", "Add stage", "Copy an existing stage", "Remove stage", and "Edit stage as JSON".

Configuration TSPlanRest Plan-Validation Deploy (Manifest) User-Validation TSDestroyRest

[new stage]

Type

Stage Name

Depends On Select...

Permalink Pipeline Actions

res

TSPlanRest  
Stage for terraspin plan operation service

TSApplyRest  
Stage for terraspin plan operation service

TSDestroyRest  
Stage for terraspin Destroy operation service

Find Artifacts From Resource (Manifest)  
Finds artifacts from a Kubernetes resource.

Entity Tags

Copy an existing stage

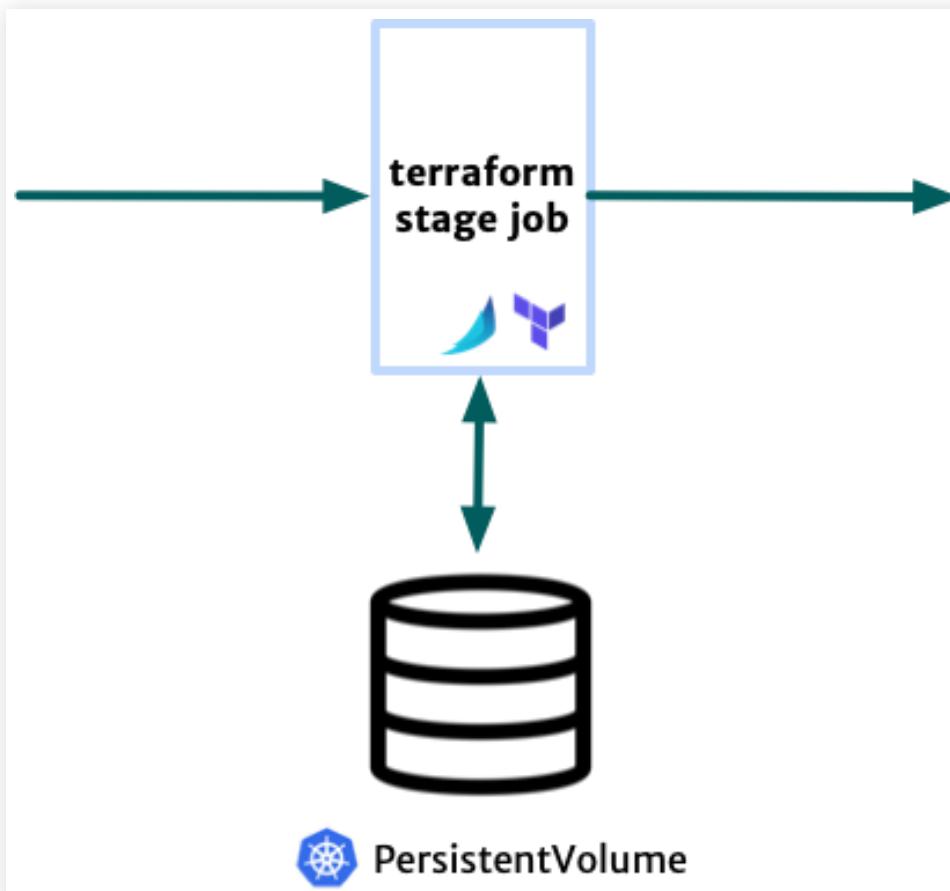
Remove stage

Edit stage as JSON

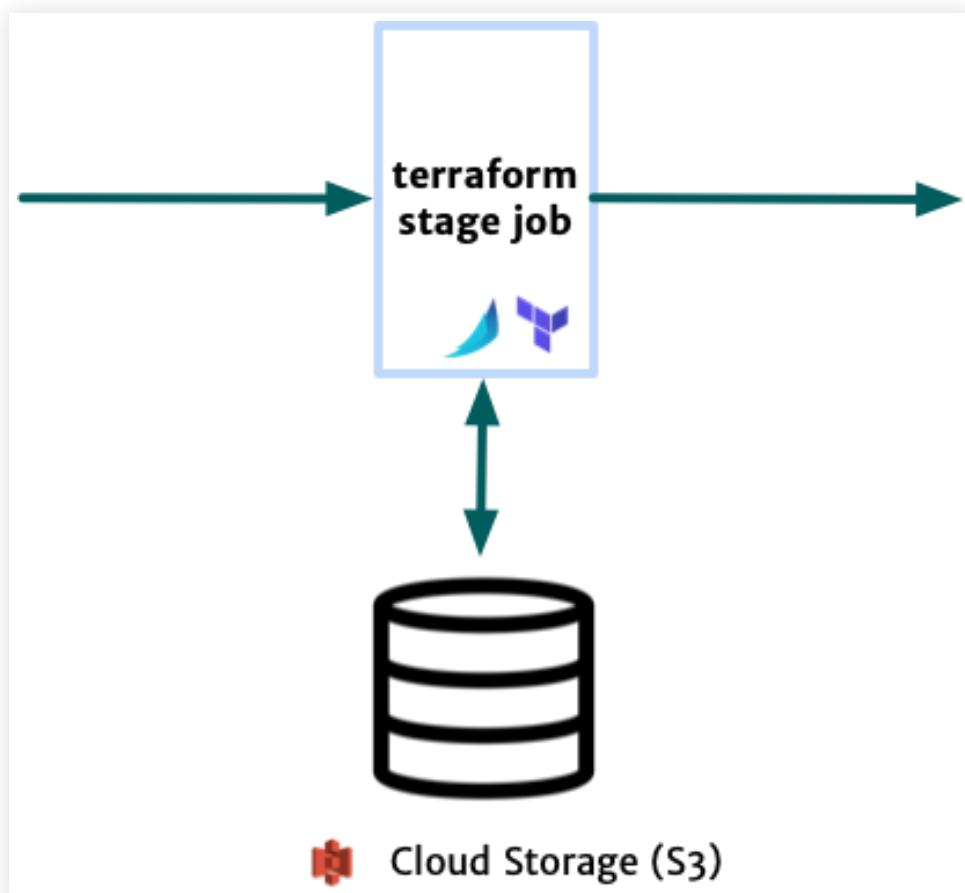
# WHERE DO WE STORE THE TERRAFORM STATE IN KUBERNETES?

- State can either or both be stored as a:
  - PersistentVolume in Kubernetes
  - Using S3 and Dynamo DB for storage

## POSSIBLE SOLUTION #1. USE A PERSISTENT VOLUME



## POSSIBLE SOLUTION #2. USE S3 BUCKET



# TRADE-OFFS IN CORPORATE ENVIRONMENT

- You certainly have trade-offs:
  - Everyone has their own Terraform instance but shares the same S3 cloud bucket
  - Run in a cloud instance that backs up in an S3 cloud bucket
  - Run as a Kubernetes Job that also backs up with S3 cloud bucket

Source: <https://learn.hashicorp.com/terraform/getting-started/build>

- COPY in
  - the *rds.tf* which is the terraform file
  - a *.sh* file that will run the job
- Make *.sh* file the entrypoint

```
FROM hashicorp/terraform:light

COPY rds.tf /
COPY run.sh /

ENTRYPOINT ["/run.sh"]
```

## .SHFILE (PART 1) RUNNING TERRAFORM

```
#!/bin/sh
terraform init
terraform apply -auto-approve
```

## .SHFILE (PART 2) SPINNAKER PROPERTY

- A technique that is certainly worth knowing is how to pass parameters from one stage to the next
- To achieve this within spinnaker, you can run the job and `echo` a string with a `SPINNAKER_PROPERTY_` prefix
- The `SPINNAKER_PROPERTY_` prefix can be used by Spinnaker to carry these values to other stages

## .SHFILE (PART 3) TERRAFORM OUTPUT

```
echo "SPINNAKER_PROPERTY_DB_ADDRESS=$(terraform output db_address | tr -d \"")"  
echo "SPINNAKER_PROPERTY_DB_NAME=$(terraform output db_name | tr -d \"")"  
echo "SPINNAKER_PROPERTY_DB_SECRET=$(terraform output db_secret | tr -d \"")"  
echo "SPINNAKER_PROPERTY_DB_USERNAME=$(terraform output db_username | tr -d \"")"
```

## BETTER OPTIONS? HASHICORP VAULT

- If your question is whether transferring `db_secrets` is safe, using `echo`, answer is likely not.
- This is where Hashicorp Vault can perform a particular role.

# OVERVIEW OF HASHICORP VAULT

- Integrate with external systems using pluggable components, called:
  - Secrets engines
  - Authentication methods
- Purpose of those components is to manage and protect your secrets in dynamic infrastructure
- (e.g. database credentials, passwords, API keys)
- Server/Client Paradigm

## HOW DO SECRETS GET STORED?

- Secrets are stored on the server
- Information is encrypted in storage
- Vault itself cannot read the information without Unsealed Keys

# EXAMPLE OF A SECRET

## Storing a Secret

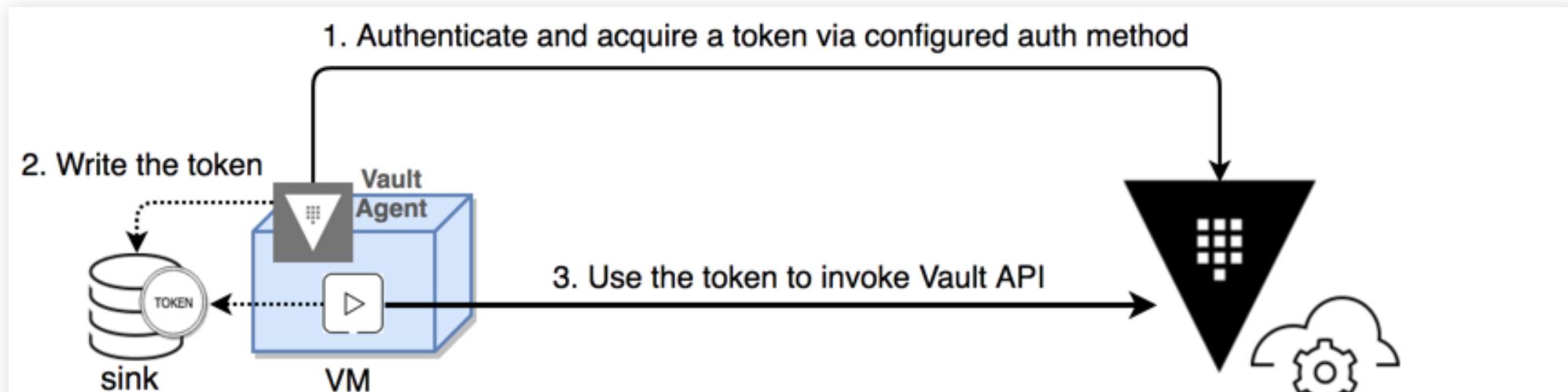
```
$ vault kv put secret/hello foo=world excited=yes
```

## Retrieving a Secret

```
$ vault kv get -field=excited secret/hello
```

- Vault Agent Sidecar

- Pods in Kubernetes Run and Agent Sidecar that is provided by an injector
- The sidecar negotiates with the Vault Server to retrieve Secrets
- Secrets are stored in a file for easy retrieval in an application



## SOURCES FOR VAULT AGENT

<https://www.youtube.com/watch?v=R3EYd9YnShU&t=2787s>

<https://github.com/jweissig/vault-k8s-sidecar-demo>

# WHAT DOES THIS MEAN WITH SPINNAKER?

In Kubernetes Job:

- Run Terraform to create infrastructure
- Terraform in the form of output generates secrets (e.g. database)
- Use Sidecar Authentication to generate the secrets to the database.

In Kubernetes App:

- Use Sidecar Authentication to retrieve the secrets to the file system of your microservice
- Agent Sidecar will place secrets conveniently in a file system of your microservice

# **LAB 5: TERRAFORM AND SPINNAKER**

AWS

# SPINNAKER PROVIDER

- Spinnaker can release artifacts in some the following Compute Services
  - **Amazon Elastic Compute Cloud (EC2)** - - Use this option, if you want to manage AWS EC2 via Spinnaker
  - **Amazon Elastic Container Service (ECS)** - - Use this option, if you want to manage containers in AWS ECS
  - **Amazon Elastic Kubernetes Service (EKS)** - Use this option, if you want to manage containers in AWS EKS. This option uses Kubernetes V2 (manifest based) Clouddriver
  - **Amazon Lambda (Lambda)** - Use this option, if you want to enable AWS Lambda support

# AWS AND IAM STRUCTURE

- Spinnaker for AWS will use AWS IAM (Identity Access Management)
- AWS requires IAM accounts set beforehand:
  - AWS Managing Account
  - AWS Managed Accounts

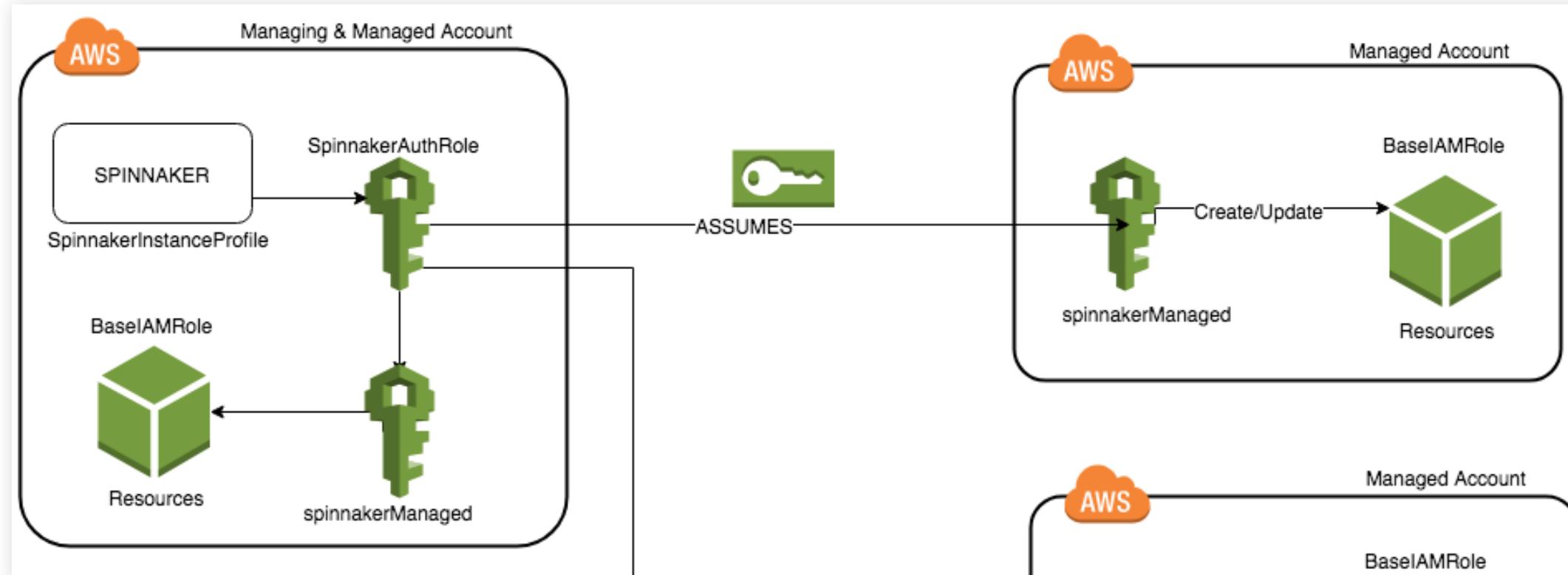
# SPINNAKER AND AWS ACCOUNTS

- Halyard configures Spinnaker to use the AWS Managing account to control the AWS Managed account(s)
- AWS Managing account assumes control of the AWS Managed account(s) through the use of AWS IAM Roles.
- By assuming a role across AWS Accounts, Spinnaker can control AWS resources from multiple AWS Managed accounts.

# SPINNAKER ACCOUNTS

- 1. AWS Managing Account** - There is always exactly one managing account. This account is what Spinnaker authenticates as and, if necessary, uses to assumes roles in the managed account(s).
- 2. AWS Managed Account** - Every AWS account that you want to modify resources in is a managed account. Managed accounts require AWS IAM policies and a trust relationship to grant AssumeRole access to the managed account(s).

Example: AWS Managing account spinnakermanaging can assume the Managed role in the accounts accountdev, accountstaging, accountprod and deploy a baked AMI in the pipeline.



## HOW TO PROVIDE ACCESS

You can grant your IAM users permission to switch to roles within your AWS account or to roles defined in other AWS accounts that you own.

## accounts

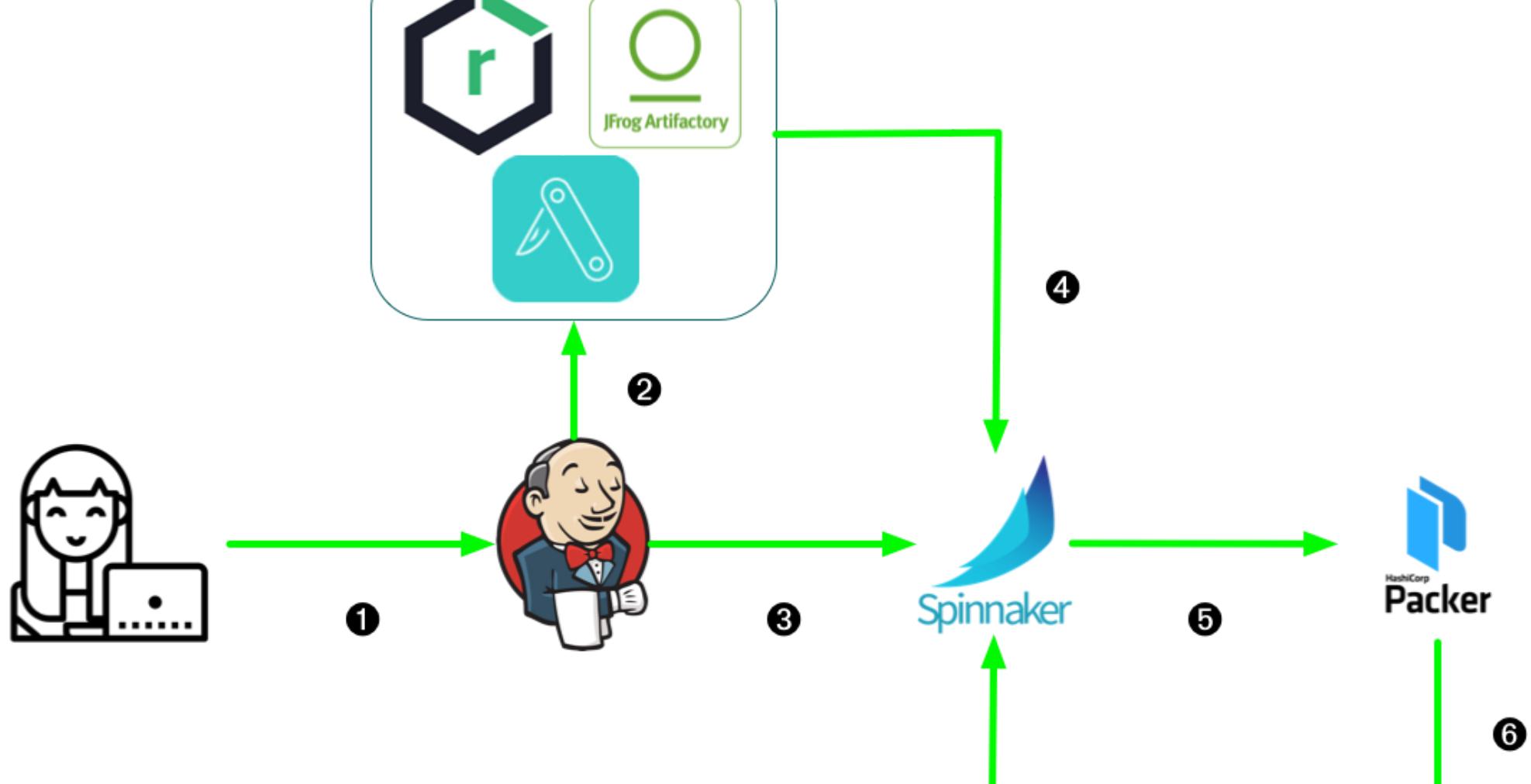
- The Managing Account manages the Managed Accounts, and we will need to provide a link between the Managing Role and the Managed Role

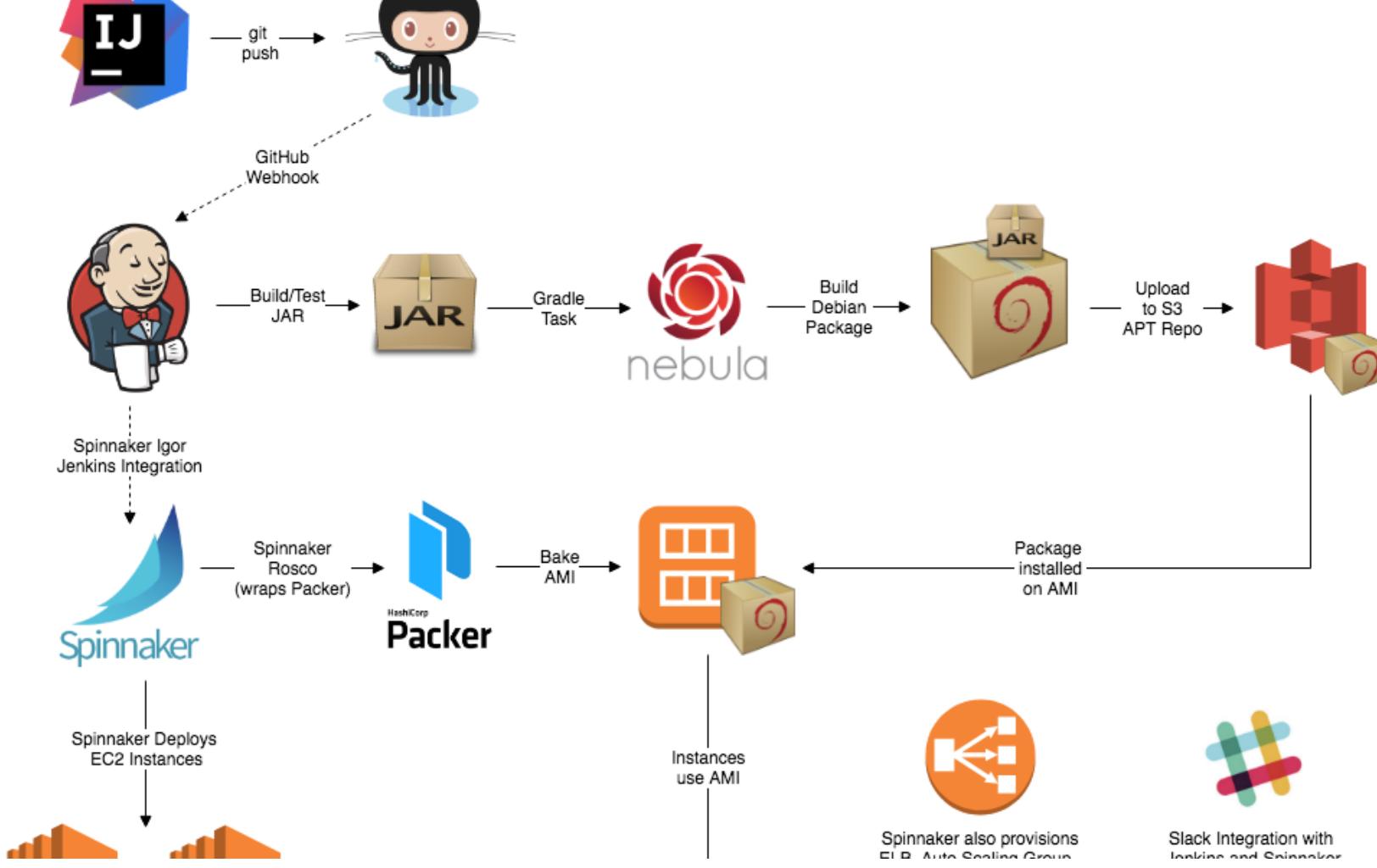
The screenshot shows the AWS Organizations console interface. At the top, there is a header with the title "Organization", a toggle switch labeled "View AWS accounts only", and a "Actions" button with a dropdown arrow. Below the header, a descriptive text states: "Organizational units (OUs) enable you to group several accounts together and administer them as a single unit instead of one at a time." The main area is titled "Organizational structure" and displays a tree view of accounts. The root account is named "Root" and has the identifier "r-6705". A sub-account named "spinnakerStaging" is shown under the Root account. This sub-account has the identifier "461826661506" and the email address "spinnakerstaging@evolutionnext.com". The "Actions" button is located in the top right corner of the main content area.

Account	Identifier	Email Address
Root	r-6705	
spinnakerStaging	461826661506	spinnakerstaging@evolutionnext.com

- You can create multiple accounts in your "organization"
- Your company may have one or more organizations
- Your team may have their own "account" where your jobs would run
- You may have accounts for purpose, like development, staging, or production

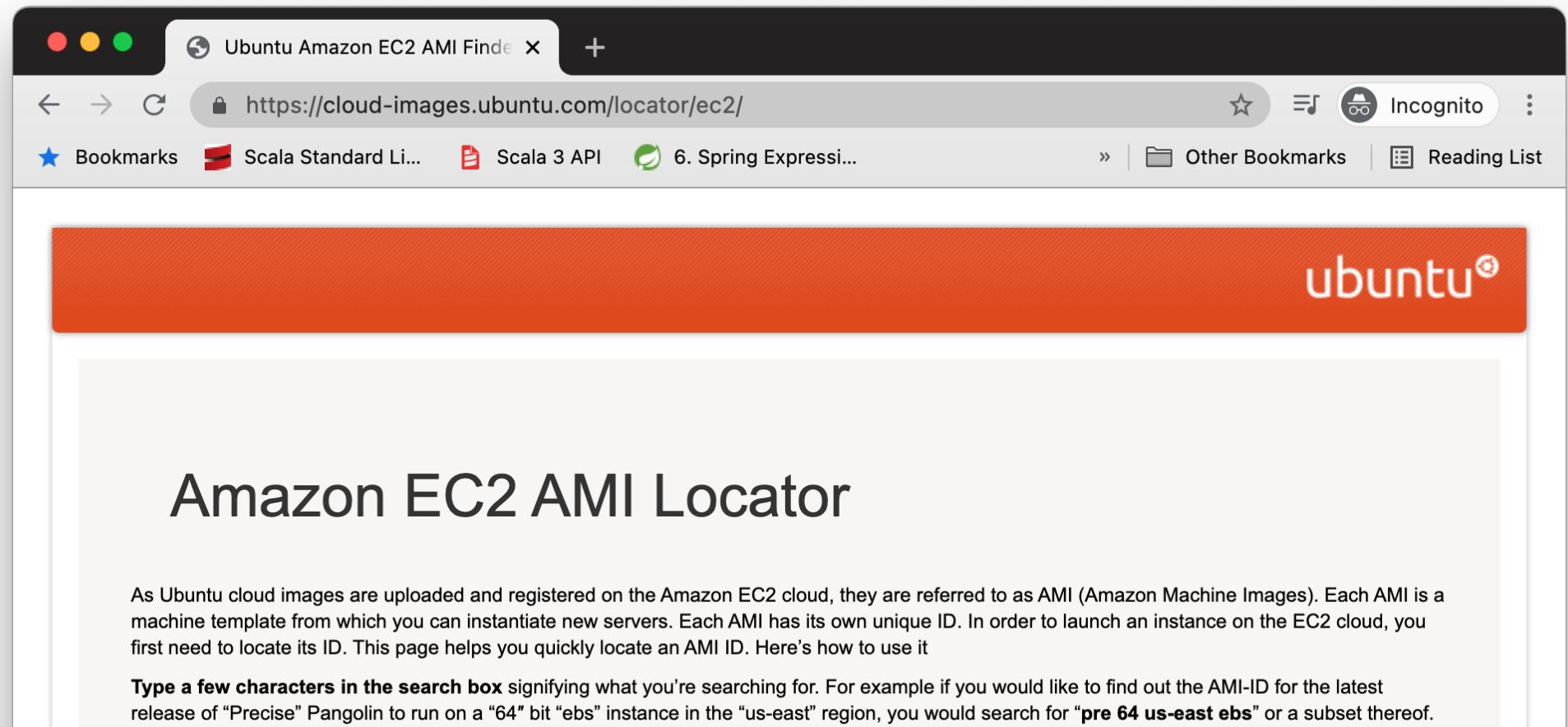
Name	Account Id
spinnakerManaging	10000000001
develop	20000000002





# BAKING

- Whereas previously, we used baking images to refer to using Helm, using an AWS Specific Provider it means making a machine image
- Jenkins will be responsible for creating a debian image (`.deb`), but not storing the debian file
- Under the hood Spinnaker uses Hashicorp Packer to build the image
- This also procures an EC2 to do the baking and creates an image as an AMI



As Ubuntu cloud images are uploaded and registered on the Amazon EC2 cloud, they are referred to as AMI (Amazon Machine Images). Each AMI is a machine template from which you can instantiate new servers. Each AMI has its own unique ID. In order to launch an instance on the EC2 cloud, you first need to locate its ID. This page helps you quickly locate an AMI ID. Here's how to use it

**Type a few characters in the search box** signifying what you're searching for. For example if you would like to find out the AMI-ID for the latest release of "Precise" Pangolin to run on a "64" bit "ebs" instance in the "us-east" region, you would search for "**pre 64 us-east ebs**" or a subset thereof.

# HANDING OFF FROM DEBIAN REPOSITORY TO JENKINS

- Jenkins is responsible for creating a debian package
- The Debian Package is created by the build tool which should be in your build script
  - Maven - [JDeb](#)
  - Gradle - [Nebula](#)

# SETTING UP A DEBIAN REPOSITORY

- Choose a Debian Repository (e.g. Artifactory, Nexus, Aptly)
- Set up Jenkins to create the debian package and install it to your Debian Repository

# SETTING UP ROSCO TO HANDLE YOUR DEBIAN REPOSITORY

- Baking means turning a Linux AMI + Debian Files + Debian Dependencies and converting it to a new AMI
- Rosco is the Microservice of Spinnaker in charge of performing that operation
- Rosco needs to be configured by Spinnaker Administrator using Halyard or other means

- This is different from Kubernetes-based jobs
- Select the kind of load balancer you wish to use, ALB, NLB, or Classic

### Select Type of Load Balancer

X

**Application  
(ALB)**

Highly configurable, application-focused balancer. HTTP and HTTPS only.

**Network  
(NLB)**

Basic, high-performance balancer with fixed IP.

**Classic  
(Legacy)**

Previous generation balancer (ELB).

Your load balancer will be named: possibleec2app?

**Account**

production

**Region**

us-west-2

**Stack** ?**Detail** ?**Availability Zones**

Automatic Availability Zone Balancing:

Enabled

# PLACING INFORMATION INTO LOAD BALANCER

- **Account** - The Account where this load balancer will be created
- **Region** - What cloud region will this load balancer go into.
- **Stack** - Naming to create vertical scaling
- **Detail** - Naming to describe the name of the deployment (Should be less than 32 characters)
- **Availability Zones** - Select which zones where the deployments should be applied
- **VPC Subnet** - Each VPC (Virtual Private Cloud) has its own subnets, subnets that can public or private, typically for routers this will be public

# CLOUD COMPONENT ARCHITECTURE

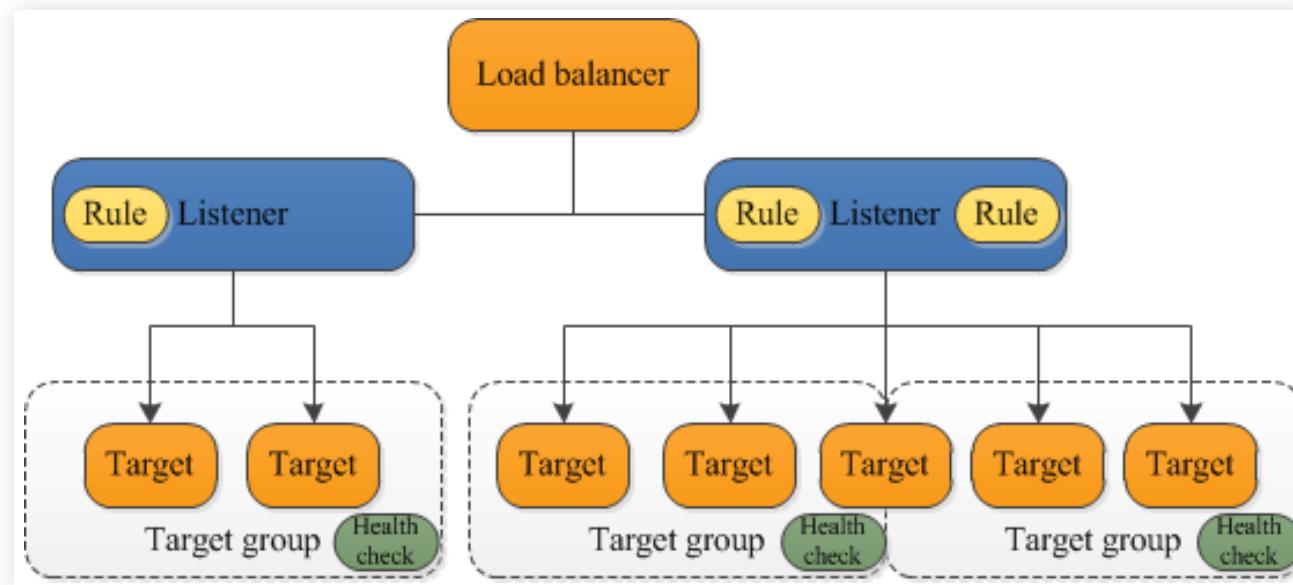


Figure 8. Component Architecture from AWS

Source:

<https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html>

# LISTENERS IN AWS

- Before you start using your Application Load Balancer, you must add one or more listeners.
- A listener is a process that checks for connection requests, using the protocol and port that you configure.
- The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

Source: <https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-listeners.html>

# TARGET GROUPS IN AWS

- Each target group is used to route requests to one or more registered targets.
- When you create each listener rule, you specify a target group and conditions.
- When a rule condition is met, traffic is forwarded to the corresponding target group.
- You can create different target groups for different types of requests.
- For example, create one target group for general requests and other target groups for requests to the microservices for your application.

Source: <https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-target-groups.html>

# **SETTING UP FIREWALL CONFIGURATION**

## Location

Your firewall will be named: possibleec2app 

**Account**

evnx

**Regions**

us-west-2

**Stack**

**Detail**

**Description  
(required)**

# FIREWALL LOCATION FIELDS

- **Account** - Which Account in AWS will be holding this account
- **Region** - Which Region do we use
- **Stack** - Naming to create vertical scaling
- **Detail** - Naming to describe the name of the deployment (Should be less than 32 characters)
- **Description** - Description of the firewall
- **VPC** - What VPC will this apply

## Ingress

**i** IP range rules can only be edited through the AWS Console.

Firewall	Protocol	Start Port	End Port
<input type="text"/>	TCP	7001	7001
<a href="#">Select from a different account or VPC</a>			
<a href="#"> Add new Firewall Rule</a>			

Firewalls last refreshed 2021-04-17 13:10:10 PDT

## FIREWALL INGRESS FIELDS

- **Firewall** which firewall do you will to apply additional rules
- **Start Port** which port do we start
- **End Port** which port do we end

## CREATING AN AWS PIPELINE

- Creating an AWS Pipeline is the same as creating a Kubernetes based Pipeline
- The only difference is that you need to use a Jenkins trigger to get the debian file

## Automated Triggers

Type	Jenkins	x ▾	Remove trigger
Controller	jenkins-master	x ▾	
Job	Select a job...	o ▾	
Property File	spinnaker.properties		
Artifact Constraints	Select or define an artifact...	▼	
<input checked="" type="checkbox"/> Trigger Enabled			

Add Trigger

# CREATE A BAKING STAGE

Bake Configuration

<b>Regions</b>	<input checked="" type="checkbox"/> us-west-2
<b>Skip Region Detection</b> ⓘ	<input type="checkbox"/> Only bake explicitly selected regions
<b>Package</b> ⓘ	spinnaker-study
<b>Base OS</b>	<input type="radio"/> xenial ⓘ <input checked="" type="radio"/> bionic ⓘ
<b>VM Type</b>	<input checked="" type="radio"/> HVM <input type="radio"/> PV
<b>Base Label</b>	<input checked="" type="radio"/> release <input type="radio"/> candidate <input type="radio"/> previous <input type="radio"/> unstable
<b>Rebake</b>	<input type="checkbox"/> Rebake image without regard to the status of any existing bake
<input type="checkbox"/> Show Advanced Options	

## HVM VS PV

- AMIs offer a choice of virtualization type: *PV* and *HVM*
- Every AWS AMI uses the Xen hypervisor on bare metal
- Xen offers two kinds of virtualization: HVM (Hardware Virtual Machine) and PV (Paravirtualization)

# HVM

- VMs running on top of their hypervisors are not aware that they are sharing processing time with other clients on the same hardware
- The host should have the capability to emulate underlying hardware for each of its guest machines.
- Provides the ability to run an operating system directly on top of a virtual machine without any modification
- Runs as if it were run on the bare-metal hardware

# PARAVIRTUALIZATION

- This technique is fast and provides near native speed in comparison to full virtualization
- Guest operating system requires some modification before everything can work.
- These modifications allow the hypervisor to export a modified version of the underlying hardware to the VMs
- This allows near native performance

# AMAZON RECOMMENDATIONS

- Amazon currently recommends users to choose HVM instead of PV.
- Ignoring their advice can have very real consequences.
- For example, in the AWS Frankfurt region, if you try to select an AWS AMI (Amazon Linux) using PV, you will be greatly restricted in your choice of instance types

Source: <https://cloudacademy.com/blog/aws-ami-hvm-vs-pv-paravirtual-amazon/>

- **Region** - Region where baking will take place
- **Skip Region Detection** - Spinnaker will detect regions to bake in from downstream deploy stages. This prevent failed deploys from accidentally missed regions during the bake process. This will disable
- **Package** - This should *be inline with the name of the debian file* that you created
- **Base OS** - What Debian Repository should we look for the application, established in Spinnaker configuration
- **VM Type** - HVM or PV
- **Rebake** - Do we need to rebake images, even if already baked in subsequent builds
- **Base Label** - What label should be applied to the baked AMI· release candidate previous

# ADVANCED BAKING STAGE

Show Advanced Options

Template File Name ⓘ

Extended Attributes ⓘ

Key	Value	Actions
<a href="#">+ Add Extended Attribute</a>		

Var File Name ⓘ

Base Name

Base AMI ⓘ

AMI Name ⓘ

AMI Suffix ⓘ

## BAKING ADVANCED FIELDS

- **Template File Name** - Which packer template do you wish to use, overriding the one that is configured by administrators
- **Extended Attributes** - Packer accepts other variables in case you have your template
- **Var File Name** - If you have an external file that stores variable use this field
- **Base Name** - This would override the base name that you would like to use
- **Base AMI** - This would override the Base OS
- **AMI Name** - Override the full name default of the baked image
- **AMI Suffix** - Override the suffix date format

# CREATE A DEPLOYMENT STAGE

When creating a server group, click add server group

Deploy Configuration

Provider	Account	Cluster	Region	Subnet	Strategy	Capacity	Instance Type	Actions
aws	EVNX	possibleec2app	us-west-2	ec2-subnet	highlander	Current Server Group	t2.micro	 

 Add server group

## Basic Settings

**Account**

evnx



**Region**

us-west-2



**VPC Subnet** ⓘ

ec2-subnet



**Stack** ⓘ

**Detail** ⓘ

**Traffic** ⓘ

Send client requests to new instances

**Strategy** ⓘ

None



Your server group will be in the cluster:

- **Account** - which account do you wish to deploy
- **Region** - which region will we apply
- **VPC Subnet** - which subnet to apply, typically this would be in a private subnet
- **Stack** - Naming to create vertical scaling
- **Detail** - Naming to describe the name of the deployment (Should be less than 32 characters)
- **Traffic** - Do we wish to start sending traffic out to the new location
- **Strategy** - What strategies do you wish to use on the deployment, some different one as opposed to Kubernetes

# LOAD BALANCERS

## Load Balancers

**Target Groups** ?

Select...

**Classic Load Balancers** ?

No load balancers found in the selected account/region/VPC

If you are looking for a load balancer or target group from a different application,

[click here](#) to load all load balancers.

## LOAD BALANCERS FIELDS

- **Target Groups** - which target groups you wish to send the information
- **Classic Load Balancers** - not particularly requires since we aren't using Classic Load Balancing

# FIREWALLS

## Firewalls

### Firewalls

Select...

default (sg-09270608f466a1006)

external-webgroup (sg-0eed13c778b92f279)

## FIREWALL FIELDS

- **Firewalls** - Select the firewalls you wish to use for your deployment

# INSTANCE TYPES

## Instance Type

This application is:



**General  
Purpose**



**High  
Memory**



**Micro  
Utility**



**\* Custom  
Type**

Select an instance type...



## INSTANCE TYPE FIELDS

- Select what type of instance you wish to deploy
- Varieties for AWS are available

- Select the capacity of instances you wish to deploy

## Capacity

Sets min, max, and desired instance counts to the same value.

To allow true auto-scaling, use the [Advanced Mode](#).

### Number of Instances

Num	<input type="text" value="\${...}"/>
1	

Consider deployment successful when  percent of instances are healthy.

- If automatic, you can which zones to deploy

## ADVANCED SETTINGS

- There are plenty of Advanced Setting when deploying, it would take a lot of time to go through each one.
- The important one is which key to use, so that an ssh connection can be made to the EC2 instance.
- Once that has been selected, the Add button should be active

# TAGS

You have the ability to create optional tags to go with the deployment you wish to make

**Tags (optional) ?**

Key	Value	
Name	mild-temper-microservice	
 Add Field		

# **LAB 6: AWS PROVIDER**

# AGENDA

## *Our mandatory goals for this class:*

- Become familiar with Spinnaker and it's role
- Understand the UI of Spinnaker
- Understand an Application, Pipeline, and Stage
- Purpose of Spinnaker
- How to get Spinnaker within Salesforce
- Helm and Baking
- Spinnaker Expression Language
- Understand the Role of Terraform

## SOME OTHER ITEMS YOU'LL PICK UP

- Canary
- Understanding of Kubernetes
- How it works
- Debugging Tips

# CONCLUSION

- Dev-ops is hard, requires a lot of knowledge in:
  - Command line programming
  - Kubernetes
  - Prometheus
  - Multiple Languages

You will need to be creative