

Creating Dynamic Agents – Using Kubernetes Pods As Jenkins Agents

Highly Recommended

- ❖ It is essential to get rid of static agents.
- ❖ However, it would help if you also remembered that it is impossible to eliminate all the static agents.
- ❖ For instance, you cannot containerize the MacOS operation system and many more.
- ❖ Running dynamic agents makes the security teams happy, and they highly recommend it.
- ❖ It means you do not have VM instances to maintain.
- ❖ This means you are not making static infrastructure costing the company money leading to wasteful resources because even when you are not using static Jenkins agents, they are still running.
- ❖ Using dynamic agents provisioned as pods by the cluster help one to create a ROBUST, SCALABLE, EFFECTIVE, EFFICIENT & SECURE CI/CD pipeline.
- ❖ The pipeline is dynamic because the engineers do not need to bother maintaining the applications' versions.
- ❖ This is because the engineers can easily define the versions they would like to work with and select the most updated ones.

In simple words, Kubernetes is going to fry the pod/s when our job/s are ready to build.

We do not need to worry about creating an agent.

An agent called Jenkins/inbound agent will be available immediately we install KUBERNETES plugin.

This is an image for Jenkins agents using TCP or WebSockets to establish inbound connection to the Jenkins master.

This agent is powered by the Jenkins Remoting library, which version is being taken from the base Docker Agent image.

Also, you need to be aware of Java Network Launching Protocol (JNLP).

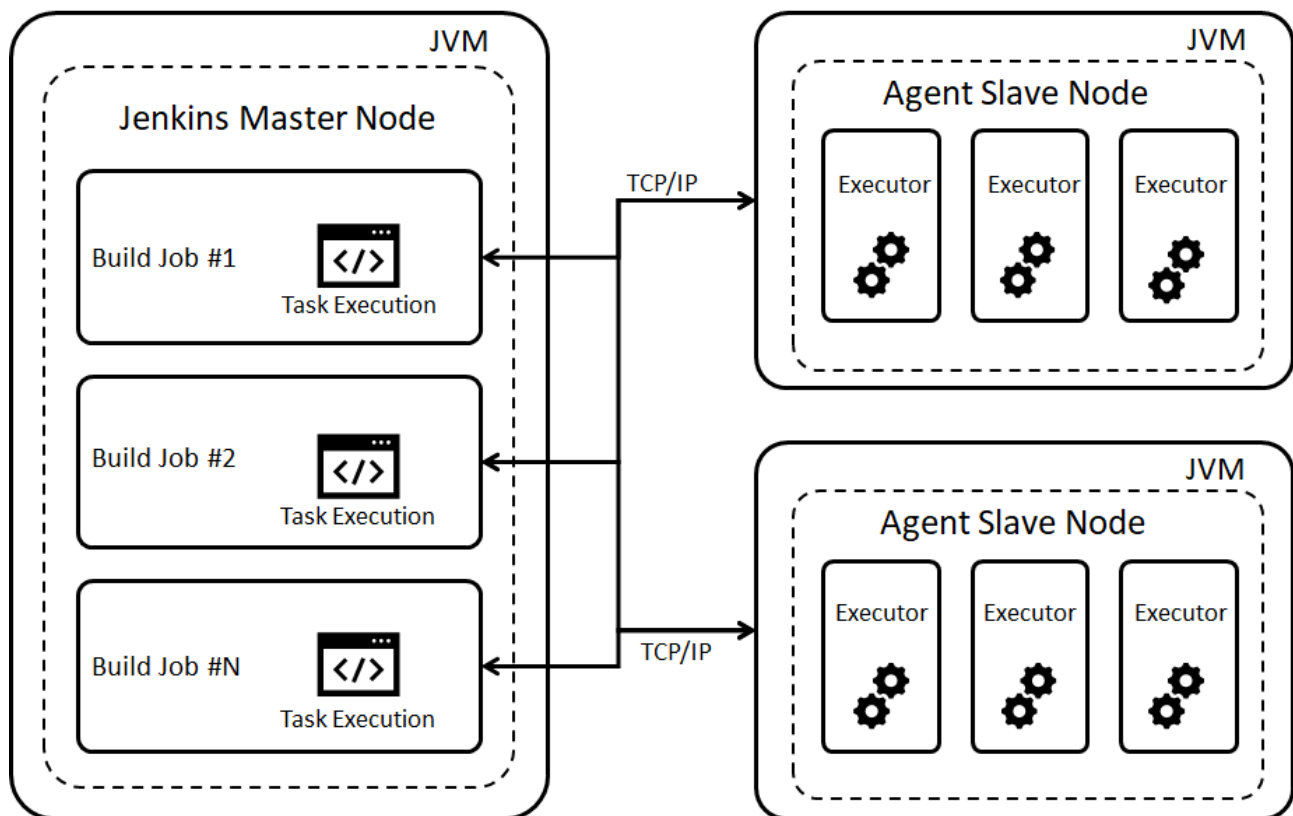
The Java Network Launch Protocol (JNLP) enables an application to be launched on a client desktop by using resources that are hosted on a remote web server.

Java Plug-in software and Java Web Start software are considered JNLP clients because they can launch remotely hosted applets and applications on a client desktop.

We can use this setup to understand the relationship between the Jenkins Master and the Slaves. Not good words to us.

We should call the Jenkins master (the control plane) and the Jenkins slaves the (Jenkins agents).

The Jenkins Control, Agent Structure



Step 1

Provision your Jenkins control using terraform/terragrunt

My jenkins control is ready



Welcome to Jenkins!

Sign in

☐ Keep me signed in

Our Jenkins Commandment

- ⇒ As DevOps engineers, we shall never build jobs in control Jenkins.
- ⇒ The work of the control Jenkins is to manage the plugins, schedule the jobs to the respective Static and Dynamic agents, manage the applications and the pipeline scripts related to the jobs, and make sure the build interfaces and logs are available to us.
- ⇒ Building with control Jenkins affects the functionality and performance of the server because it becomes overburdened with too many tasks running simultaneously.
- ⇒ It is also not secure to build with control Jenkins.

The first step before configuring anything else is to disable the control Jenkins build executors to ensure the Jenkins control will never schedule or build a job.

Steps

Click on



New Item



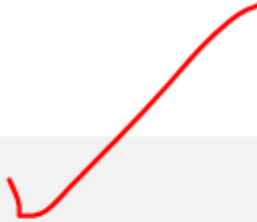
People



Build History



Manage Jenkins



My Views



Lockable Resources



New View

Next

Click on

Manage Jenkins

System Configuration



Configure System

Configure global settings and paths.



Global Tool Configuration

Configure tools, their locations and automatic installers.



Manage Nodes and Clouds

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.



Next



Back to Dashboard



Manage Jenkins



New Node



Configure Clouds



Node Monitoring

Build Queue



No builds in the queue.

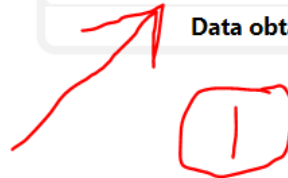
Build Executor Status





Next


Manage nodes and clouds


| S | Name ↓ | Architecture |
|---|---------------|---------------|
| | Built-In Node | Linux (amd64) |
| | Data obtained | 47 min |





 Back to List

 Status

 Configure


 Build History

 Load Statistics

 Script Console

Build Executor Status

^



Built-In Node (the Jenkins contro

Projects tied to Built-In Node

None

Next

Change the number of executors to zero and save

Number of executors ?

0



Labels ?

Usage ?

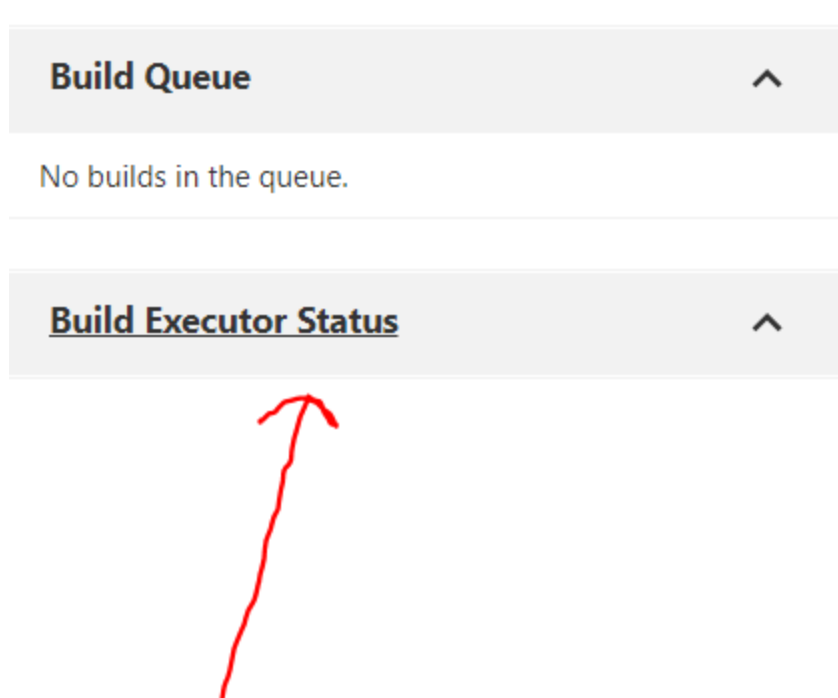
Use this node as much as possible

Node Properties

- ☐ Disable deferred wipeout on this node ?
- ☐ Environment variables

Save

Confirm no executor exists



No executor exists!!!! → you can now move to the next steps below

2. Install the Kubernetes plugin - this plugin integrates Jenkins with Kubernetes

Manage Jenkins → manage plugins → available → install without restart

3. Go to Manage Jenkins → manage nodes and clouds



Manage Nodes and Clouds

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Click on manage nodes and clouds



Back to Dashboard



Manage Jenkins



New Node



Configure Clouds



Node Monitoring

Build Queue



No builds in the queue.

Build Executor Status



⇒ Click on Configure Cloud

Configure Clouds



Kubernetes

Name ?

kubernetes

you can change Name



Kubernetes Cloud details...

Pod Templates...

Delete cloud

Add a new cloud ▾

Save


Apply

You can change the name from Kubernetes to whatever you like.

The rule of the thumb is to leave the name as Kubernetes.

⇒ Next

Configure Clouds

 **Kubernetes**
Name ?

kubernetes



Kubernetes Cloud details...

Pod Templates...

Delete cloud


Add a new cloud ▾

Save

Apply

⇒ Next

Configure Clouds

 **Kubernetes**
Name ?

kubernetes

Kubernetes URL ?

https://35.229.91.229

☐ Use Jenkins Proxy ?

Kubernetes server certificate key ?

☒ Disable https certificate check ?

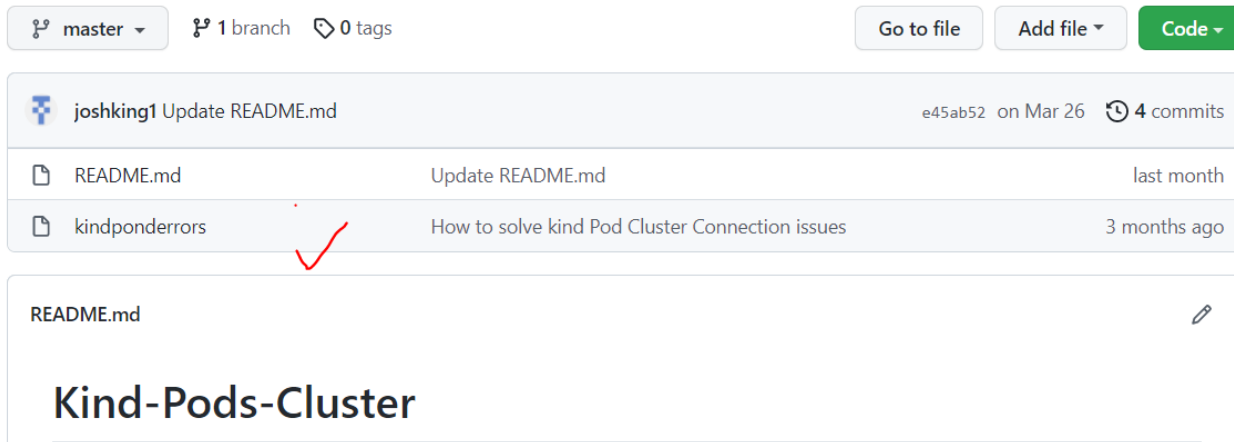
Kubernetes Namespace

jenkins

Step 1: Get the url for the cluster

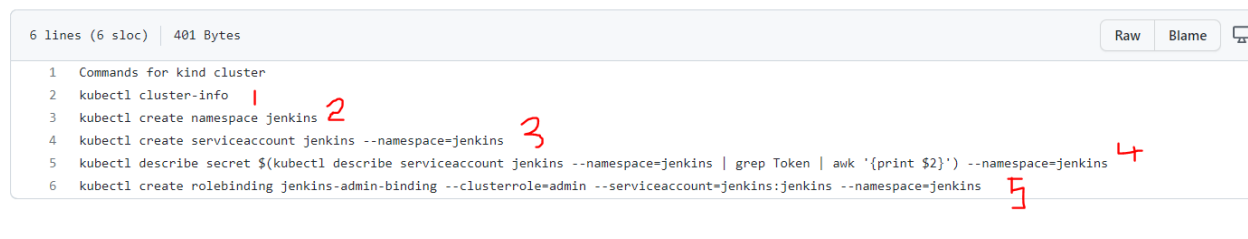
Go to ➔ <https://github.com/joshking1/Kind-Pods-Cluster.git> and fork it

You should have the following contents



Click on Kindpoderrors – Spelling (never mind). I did fix it.

You should have these commands



To get the Kubernetes URL, go to the Kubernetes cluster node terminal

For this project we are using the Google Kubernetes Cluster

Run the command number 1 in your cluster

Command: # kubectl cluster-info

Output

```
devopscow91@cloudshell:~ (boutique-appplication)$ kubectl cluster-info
Kubernetes control plane is running at https://35.229.91.229
GLBCDefaultBackend is running at https://35.229.91.229/api/v1/namespaces/kube-system/services/default-http-backend:http/proxy
KubeDNS is running at https://35.229.91.229/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://35.229.91.229/api/v1/namespaces/kube-system/services/https:metrics-server:proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Take the URL and paste it here

Kubernetes URL ?

https://35.229.91.229



☐ Use Jenkins Proxy ?

2 Kubernetes certificate

☐ Use Jenkins Proxy ?

Kubernetes server certificate key ?

☒ Disable https certificate check ?

Kubernetes Namespace

jenkins

Disable certificate check – this should never happen in production

We can feed the certificate and uncheck the disabled icon, but for now, do not worry about this → I will guide you on this when we reach this stage.

3. Create a namespace called jenkins

Kubernetes Namespace

jenkins

Command

kubectl create namespace jenkins

Output

```
devopscow91@cloudshell:~ (boutique-application)$ kubectl create namespace jenkins
namespace/jenkins created
devopscow91@cloudshell:~ (boutique-application)$ █
```

Check whether the namespace exists

4. Create a Service Account

A service account exists in a namespace within a cluster.

Let create a service account using command

```
# kubectl create serviceaccount jenkins --namespace=jenkins
```

```
devopscow91@cloudshell:~ (boutique-application)$ kubectl create serviceaccount jenkins --namespace=jenkins
serviceaccount/jenkins created
devopscow91@cloudshell:~ (boutique-application)$
```


Let us get the token generated and which is connected to this service account


```
Command: # kubectl describe secret $(kubectl describe serviceaccount jenkins --  
namespace=jenkins | grep Token | awk '{print $2}') --namespace=jenkins
```

Output

Let go to our Jenkins and test the connection

Credentials

- none - 

❌ Error testing connection https://35.229.91.229: Failure executing: GET at: https://35.229.91.229/api/v1/namespaces/jenkins/pods. Message: pods is forbidden: User "system:anonymous" cannot list resource "pods" in API group "" in the namespace "jenkins". Received status: Status(apiVersion=v1, code=403, details=StatusDetails(causes=[], group=null, kind=pods, name=null, retryAfterSeconds=null, uid=null, additionalProperties={}), kind=Status, message=pods is forbidden: User "system:anonymous" cannot list resource "pods" in API group "" in the namespace "jenkins", metadata=ListMeta(_continue=null, remainingItemCount=null, resourceVersion=null, selfLink=null, additionalProperties={}), reason=Forbidden, status=Failure, additionalProperties={}). 

Sure, the cluster does not recognize the account. System anonymous.

WHY?

A positive thing – namespace Jenkins is recognized

A negative thing – service account we created is not recognized by the Kubernetes API when we try to connect.


Solutions

1. Create a secret text and save the token generated by the service account.


Click on ADD


Credentials

- none - 

❌ Error testing connection https://35.229.91.229: Failure executing: GET at: https://35.229.91.229/api/v1/namespaces/jenkins/pods. Message: pods is forbidden: User "system:anonymous" cannot list resource "pods" in API group "" in the namespace "jenkins". Received status: Status(apiVersion=v1, code=403, details=StatusDetails(causes=[], group=null, kind=pods, name=null, retryAfterSeconds=null, uid=null, additionalProperties={}), kind=Status, message=pods is forbidden: User "system:anonymous" cannot list resource "pods" in API group "" in the namespace "jenkins", metadata=ListMeta(_continue=null, remainingItemCount=null, resourceVersion=null, selfLink=null, additionalProperties={}), reason=Forbidden, status=Failure, additionalProperties={}). 

Nex ➡

 **Jenkins Credentials Provider: Jenkins**

 **Add Credentials**

Domain
Global credentials (unrestricted) ▼

Kind
Secret text ✓

Scope ?
Global (Jenkins, nodes, items, all child items, etc) ▼

Secret
↓ Paste

ID ?

Next ➔

=> Paste the secret generated

=> Create a special ID

=> Add description

=> Add

Global credentials (unrestricted) ▼

Kind
Secret text ▼

Scope ?
Global (Jenkins, nodes, items, all child items, etc) ▼

Secret
.....

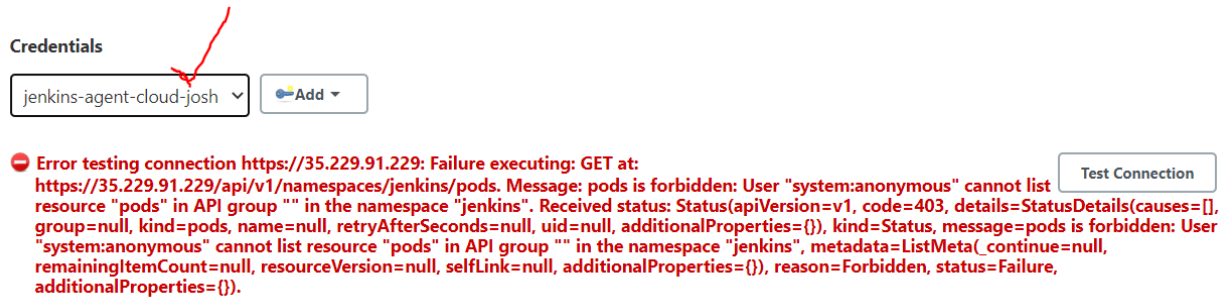
ID ?
jenkins-agent-cloud-josh

Description ?
jenkins-agent-cloud-josh

✓ Add Cancel

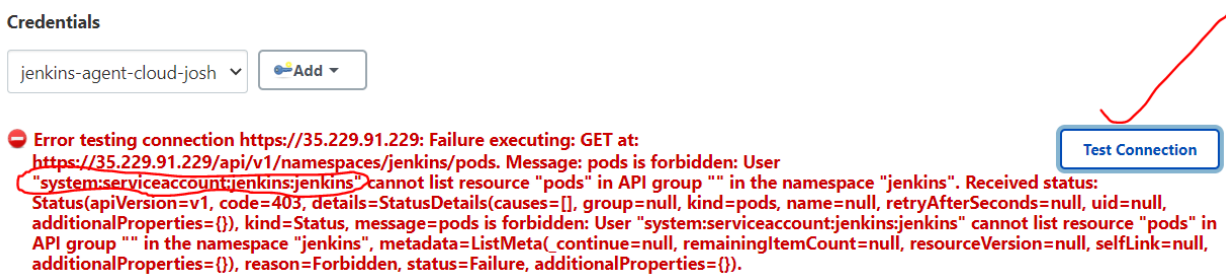
Next ➔

⇒ Select the name of the token we just created.



The screenshot shows the Jenkins 'Credentials' page. A dropdown menu is set to 'jenkins-agent-cloud-josh'. Below it, a red error message is displayed: 'Error testing connection https://35.229.91.229: Failure executing: GET at: https://35.229.91.229/api/v1/namespaces/jenkins/pods. Message: pods is forbidden: User "system:anonymous" cannot list resource "pods" in API group "" in the namespace "jenkins". Received status: Status(apiVersion=v1, code=403, details=StatusDetails(causes=[], group=null, kind=pods, name=null, retryAfterSeconds=null, uid=null, additionalProperties={}), kind=Status, message=pods is forbidden: User "system:anonymous" cannot list resource "pods" in API group "" in the namespace "jenkins", metadata=ListMeta(continue=null, remainingItemCount=null, resourceVersion=null, selfLink=null, additionalProperties={}), reason=Forbidden, status=Failure, additionalProperties={}).' A 'Test Connection' button is visible on the right.

Next ➔ Run the test connection



The screenshot shows the same Jenkins 'Credentials' page. The error message is the same as before, but the text 'system:serviceaccount:jenkins:jenkins' is circled in red. A red checkmark is drawn over the 'Test Connection' button on the right.

Great, we are no longer anonymous. The Kubernetes API have recognized the services account.

Still have a problem ➔ the Service account is forbidden

WHY?

This is because we do not have ROLE (permission created) and ROLEBINDING (attach the permission to the created service account).

Next ➔ Create Role and Role binding

#5 Create rbac (Role based access control)

There are four objects that are used by the Kubernetes API (application programming Interface)

Role – permission

Rolebinding – attaches the permission to the service account created

Both role and rolebinding are namespace oriented and are used to give machines and processes access to resources existing a namespace in the cluster.

Do not forget service account is namespace oriented

Let Create a role and a rolebinding

I am going to teach you how to use YAML file to create the roles and rolebinding instead of using commands when we meet.

Command

```
kubectl create rolebinding jenkins-admin-binding --role=admin --serviceaccount=jenkins:jenkins --namespace=jenkins
```

In this role, we are doing the following:

1. Creating a role called admin (access to all the resources in the cluster namespace)
2. Creating a rolebinding (attach the permission created to the services account)


Output

```
devopscow91@cloudshell:~ (boutique-application)$ kubectl create rolebinding jenkins-admin-binding --clusterrole=admin --serviceaccount=jenkins:jenkins --namespace=jenkins
rolebinding.rbac.authorization.k8s.io/jenkins-admin-binding created
devopscow91@cloudshell:~ (boutique-application)$
```

Next ➔

Let test the connection

Credentials

jenkins-agent-cloud-josh 

Connected to Kubernetes v1.21.10-gke.2000



Next ➔

Save

Pod Retention...

Max connections to Kubernetes API ?

32

Seconds to wait for pod to be running ?

600

Advanced...

Pod Templates...

Delete cloud

Add a new cloud ▾

Save Apply

We are ready to create a Dynamic Pipeline

⇒ Go to ➔ manage Jenkins ➔ New Item

Dashboard ▾ ▶



New Item



People



Build History



Manage Jenkins



My Views



Lockable Resources



New View

Build Queue



No builds in the queue.

Build Executor Status



Create a name for your pipeline


⇒ Select pipeline


⇒ ok


Enter an item name


jenkins-plugin-kubernetes-started by josh


» Required field

 **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

 **Folder**
A container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Familiarize yourself with build trigger

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☐ GitHub hook trigger for GITScm polling ?

☒ Poll SCM ?

Schedule ?

H/60 * * * *

Would last have run at Monday, April 25, 2022 at 8:46:49 PM Coordinated Universal Time; would next run at Monday, April 25, 2022 at 9:46:49 PM Coordinated Universal Time.

Build trigger will automatically run a build

In our case, the build will be triggered after every 60 minutes

DevOps Commandment #2

We only run declarative jenkins pipeline script as DevOps engineer.

Never run a scripted jenkins pipeline. This kind of a pipeline has a steep learning curve and companies do not like it.

In addition to that, most tools such as terraform, terragrunt, ansible, and Kubernetes are all based on declarative languages.

Pipeline to copy and paste on the pipeline section

```
pipeline {
  tools {
    maven 'Maven-3.8.4'
  }
  agent {
    kubernetes {
      yaml '''
      apiVersion: v1
      kind: Pod
      spec:
        containers:
          - name: maven
            image: maven:alpine
            command:
              - cat
            tty: true
          - name: node
            image: node:16-alpine3.12
            command:
              - cat
            tty: true
      '''
    }
  }
}
```

```
    ""
  }
}
stages {
  stage('Run maven') {
    steps {
      container('maven') {
        sh 'mvn -version'
        sh 'echo Hello World > hello.txt'
        sh 'ls -last'
      }
    }
  }
  stage ("Git Checkout") {
    steps {
      git credentialsId: 'GIT_HUB_CREDENTIALS', url:
'https://github.com/joshking1/myRepoForJavaApp.git'
    }
  }
  stage ("Maven Clean Build") {
    steps {
      sh 'mvn clean install -f MyWebApp/pom.xml'
    }
  }
}
}
```

Pipeline

Definition

Pipeline script

Script ?

```
1 pipeline {
2   tools {
3     maven 'Maven-3.8.4'
4   }
5   agent {
6     kubernetes {
7       yaml '''
8         apiVersion: v1
9         kind: Pod
10        spec:
11          containers:
12            - name: maven
13              image: maven:alpine
14              command:
15                - cat
16              tty: true
17            - name: node
```

try sample Pipeline...

Pipeline

Definition

Pipeline script

Script ?

```
18       image: node:16-alpine3.12
19       command:
20         - cat
21       tty: true
22     ...
23   }
24 }
25 stages {
26   stage('Run maven') {
27     steps {
28       container('maven') {
29         sh 'mvn -version'
30         sh 'echo Hello World > hello.txt'
31         sh 'ls -last'
32       }
33     }
34 }
```

try sample Pipeline...

Pipeline

Definition

Pipeline script

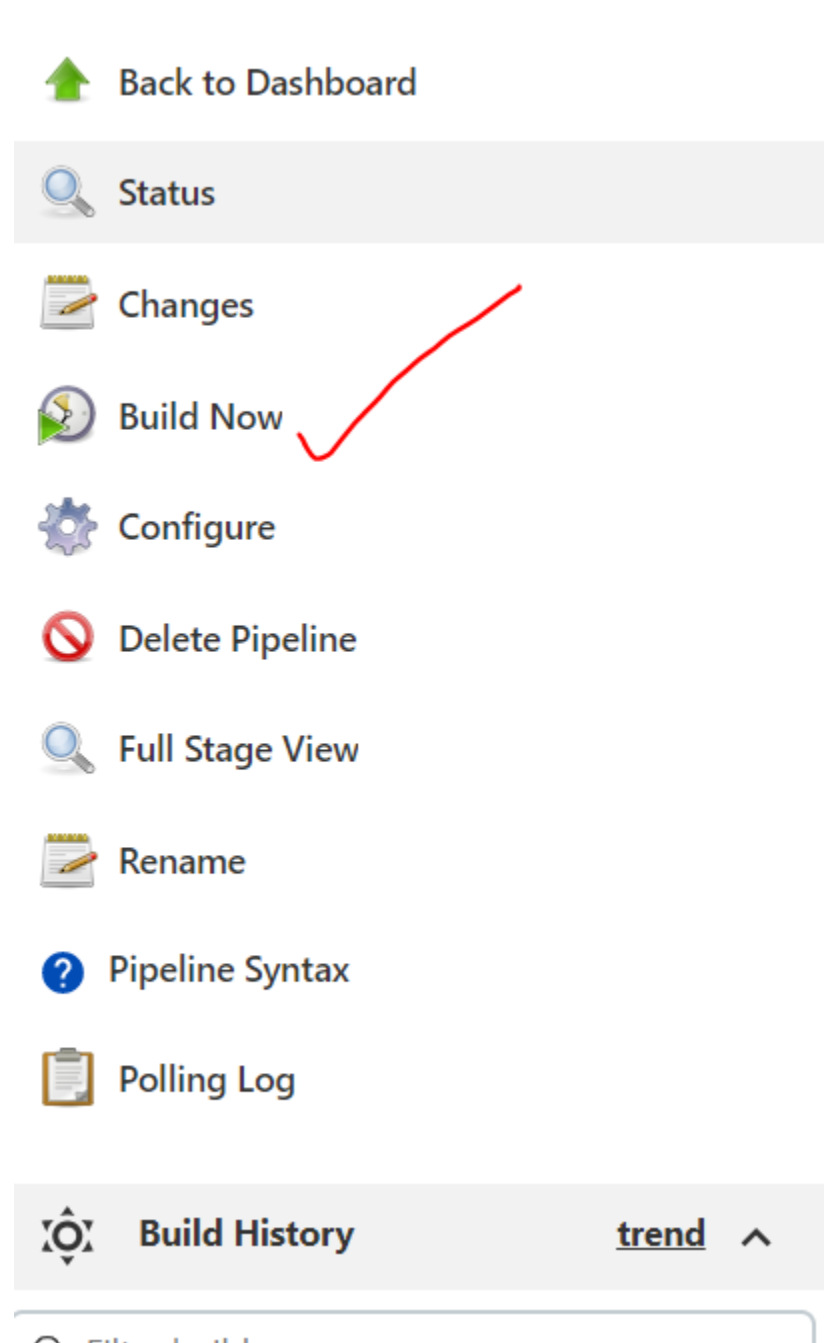
Script ?

```
30       sh 'echo Hello World > hello.txt'
31       sh 'ls -last'
32     }
33   }
34 }
35 stage ("Git Checkout") {
36   steps {
37     git credentialsId: 'GIT_HUB_CREDENTIALS', url: 'https://github.com/joshking1/myRepoForJavaApp.git'
38   }
39 }
40 stage ("Maven Clean Build") {
41   steps {
42     sh 'mvn clean install -f MyWebApp/pom.xml'
43   }
44 }
45 }
46 }
```

try sample Pipeline...

Save

Now let us build



Let see the magic of dynamic agents provisioned as a pod by the cluster

Console Output

Progress: 

```
Started by user 10
[Pipeline] Start of Pipeline
[Pipeline] podTemplate
[Pipeline] {
[Pipeline] node
Created Pod: kubernetes jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1
[Normal][jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1][Scheduled] Successfully assigned jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1 to gke-jessica-cluster-default-pool-493a908a-12vm
[Normal][jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1][Pulled] Container image "maven:alpine" already present on machine
[Normal][jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1][Created] Created container maven
[Normal][jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1][Started] Started container maven
[Normal][jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1][Pulled] Container image "node:16-alpine3.12" already present on machine
[Normal][jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1][Created] Created container node
[Normal][jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1][Started] Started container node
[Normal][jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1][Pulled] Container image "jenkins/inbound-agent:4.11-1-jdk11" already present on machine
[Normal][jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1][Created] Created container jnlp
[Normal][jenkins/jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1][Started] Started container jnlp
Agent jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1 is provisioned from template jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1
---
apiVersion: "v1"
kind: "Pod"
metadata:
  annotations:
    buildUrl: "http://34.75.159.27:8080/job/jenkins-plugin-kubernetes-started%20by%20josh/1/"
    runUrl: "job/jenkins-plugin-kubernetes-started%20by%20josh/1/"
  labels:
    jenkins: "slave"
    jenkins/label-digest: "0d4c061b659f2017c827d61d60b10d4747a680f6"
```

This is the kind of output you should view







You should have two nodes.

One for the build in node for Jenkins (permanent)

The second one is a jenkins/inbound-agent created by Kubernetes when the job started (Fried on demand and should disappear after the build is completed)

Manage nodes and clouds

Refresh status

| S | Name ↓ | Architecture | Clock Difference | Free Disk Space | Free Swap Space | Free Temp Space | Response Time |
|---|---|---------------|------------------|-----------------|---|-----------------|---|
|  | Built-In Node | Linux (amd64) | In sync | 15.30 GB |  0 B | 15.30 GB | 0ms  |
|  | jenkins-plugin-kubernetes-started-by-josh-1-jg0tb-154pk-52kp1 | Linux (amd64) | In sync | 90.89 GB |  0 B | 90.89 GB | 688ms  |
| Data obtained | | 26 sec | 26 sec | 26 sec | 26 sec | 26 sec | 26 sec |

Let confirm the agent has disappeared

⇒ go to executor

Go ➔ Dashboard

Dashboard ▾

New Item *Click*

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue ^

No builds in the queue.


Build Executor Status ^

| S | W | Name ↓ | Last Success | Last Failure | Last Duration |
|---|---|---|-----------------|-----------------|----------------|
| ✓ | 🔧 | docker-agent-1234 | 11 hr #19 | 11 hr #18 | 48 sec ▶ |
| ✗ | 🔧 | docker-pipeline | 1 day 0 hr #4 | N/A | 37 sec ▶ |
| ✓ | 🔧 | docker-pipeline-2 | 8 hr 52 min #17 | 10 hr #5 | 3 min 22 sec ▶ |
| ✓ | 🔧 | jenkins-jenkins-jenkins | 2 days 19 hr #1 | N/A | 24 sec ▶ |
| ✓ | 🔧 | jenkins-plugin-kubernetes | 2 days 19 hr #6 | 2 days 19 hr #5 | 21 sec ▶ |
| ✓ | 🔧 | jenkins-plugin-kubernetes-3 | 2 days 19 hr #1 | N/A | 36 sec ▶ |
| ✓ | 🔧 | jenkins-plugin-kubernetes-started by josh | 1 hr 1 min #1 | N/A | 1 min 13 sec ▶ |

Click

You can add more stages if you want

Pipeline jenkins-plugin-kubernetes-started by josh

 Recent Changes

Stage View

Average stage times:
(Average full run time: ~1min 13s)

#1 Apr 25 17:20 No Changes

| Declarative: Tool Install | Run maven | Git Checkout | Maven Clean Build |
|------------------------------|-----------|--------------|----------------------|
| 6s | 4s | 26s | 26s |
| 6s | 4s | 26s | 26s |

Permalinks

Let enjoy the **DYNAMIC Miracles** of Kubernetes

Follow Me

1. Go to ➔ Dashboard

Select the descriptive pipeline you previously ran

Look for the Build Now

Pipeline jenkins-plugin-kubernetes-started by josh

[Recent Changes](#)

Stage View
























Average stage times:
(Average full run time: ~1min 13s)

| | Declarative: Tool Install | Run maven | Git Checkout | Maven Clean Build |
|----|---------------------------|-----------|--------------|-------------------|
| #1 | 6s | 4s | 26s | 26s |

Permalinks









- Last build (#1), 1 hr 14 min ago

2. Click the Build Now 20 times and you should view jobs aligning

|  Build History | | trend  |
|---|--|---|
| <input type="text" value="Filter builds..."/> | | |
|  #31 | Apr 25, 2022, 10:42 PM  |  |
|  #30 | Apr 25, 2022, 10:42 PM  |  |
|  #29 | Apr 25, 2022, 10:42 PM  |  |
|  #28 | Apr 25, 2022, 10:42 PM  |  |
|  #27 | Apr 25, 2022, 10:42 PM  |  |
|  #26 | Apr 25, 2022, 10:42 PM  |  |
|  #25 | Apr 25, 2022, 10:42 PM  |  |
































I have 28 builds progress

3. Go to Build Executor Status

| Build Executor Status ^ | | |
|---|--------------------|---|
|  Built-In Node | | |
| jenkins-plugin-kubernetes-started by josh <div></div> | #2 |  |
| jenkins-plugin-kubernetes-started by josh <div></div> | #3 |  |
| jenkins-plugin-kubernetes-started by josh <div></div> | #4 |  |
| jenkins-plugin-kubernetes-started by josh <div></div> | #5 |  |
| jenkins-plugin-kubernetes-started by josh <div></div> | #6 |  |
| jenkins-plugin-kubernetes-started by josh <div></div> | #7 |  |
| jenkins-plugin-kubernetes-started by josh <div></div> | #8 |  |

For my case 28 agents were provisioned


Your number of the Jenkins agents might be different

| | | | | | | |
|--|---------------|---------|----------|---|----------|--|
|  Built-In Node | Linux (amd64) | In sync | 15.30 GB |  0 B | 15.30 GB | 0ms  |
|  jenkins-plugin-kubernetes-started-by-josh-10-jfftj-qkjrd-k47qg | | N/A | N/A | N/A | N/A | N/A  |
|  jenkins-plugin-kubernetes-started-by-josh-11-c30lq-rz578-j2rks | | N/A | N/A | N/A | N/A | N/A  |
|  jenkins-plugin-kubernetes-started-by-josh-2-zwdj2-bwl9k-5xdr6 | Linux (amd64) | In sync | 90.80 GB |  0 B | 90.80 GB | 1058ms  |
|  jenkins-plugin-kubernetes-started-by-josh-3-lw2wc-ms9r8-p5vts | Linux (amd64) | In sync | 90.80 GB |  0 B | 90.80 GB | 1057ms  |
|  jenkins-plugin-kubernetes-started-by-josh-4-7088f-31fv4-g21v4 | Linux (amd64) | In sync | 90.53 GB |  0 B | 90.53 GB | 1057ms  |
|  jenkins-plugin-kubernetes-started-by-josh-5-2hsch-1103l-l88rk | Linux (amd64) | In sync | 90.53 GB |  0 B | 90.53 GB | 243ms  |
|  jenkins-plugin-kubernetes-started-by-josh-6-pbw77-5fwpt-vcmvf | Linux (amd64) | In sync | 90.80 GB |  0 B | 90.80 GB | 1057ms  |
|  jenkins-plugin-kubernetes-started-by-josh-7-xcr5l-hz0v9-wt87g | Linux (amd64) | In sync | 90.80 GB |  0 B | 90.80 GB | 1056ms  |
|  jenkins-plugin-kubernetes-started-by-josh-8-z3t3r-qw8g3-61559 | Linux (amd64) | In sync | 90.53 GB |  0 B | 90.53 GB | 242ms  |
|  jenkins-plugin-kubernetes-started-by-josh-9-8s2kz-1xt3g-xzqkr | Linux (amd64) | In sync | 90.53 GB |  0 B | 90.53 GB | 601ms  |
| Data obtained | | 43 sec | 43 sec | 42 sec | 41 sec | 42 sec 43 sec |




Each job that is building is assigned a special and dynamic node.

In total you should have nodes that are equal to the number of jobs building simultaneously.

If you go to your Google Kubernetes cluster, and they click on Workload, you should be able to view pods running and terminating at the same time.

 Kubernetes Engine

Workloads

 REFRESH
 DEPLOY
 DELETE

OPERATION

Clusters

Workloads

Services & Ingress

Applications

Secrets and ConfigMaps

Storage

Object Browser

Migrate to containers

Backup for GKE

Config Management

Cluster

Namespace

RESET

SAVE

Workloads are deployable units of computing that can be created and managed in a cluster.

OVERVIEW

COST OPTIMISATION

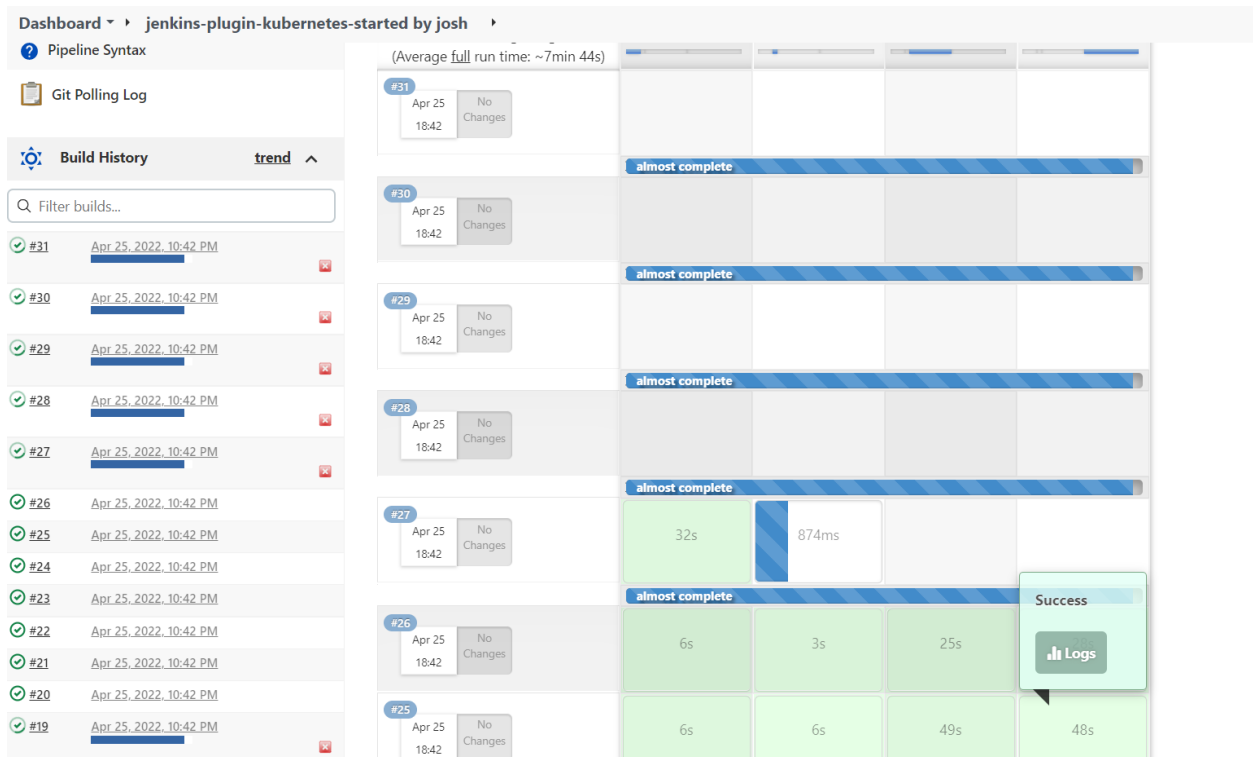
Filter

Is system object: False

Filter workloads

| <input type="checkbox"/> | Name ↑ | Status | Type | Pods | Namespace | Cluster |
|--------------------------|--|-------------|------|------|-----------|-----------------|
| <input type="checkbox"/> | jenkins-plugin-kubernetes-started-by-josh-12-l0zk5-p1rk2-cxpx6 | Running | Pod | 1/1 | jenkins | jessica-cluster |
| <input type="checkbox"/> | jenkins-plugin-kubernetes-started-by-josh-13-w1k8t-kjvdz-q7mjl | Running | Pod | 1/1 | jenkins | jessica-cluster |
| <input type="checkbox"/> | jenkins-plugin-kubernetes-started-by-josh-14-jghct-0g695-thz7m | Running | Pod | 1/1 | jenkins | jessica-cluster |
| <input type="checkbox"/> | jenkins-plugin-kubernetes-started-by-josh-15-pf4nf-xnp7w-nmh05 | Running | Pod | 1/1 | jenkins | jessica-cluster |
| <input type="checkbox"/> | jenkins-plugin-kubernetes-started-by-josh-16-mvv5b-w9jct-5qdwk | Running | Pod | 1/1 | jenkins | jessica-cluster |
| <input type="checkbox"/> | jenkins-plugin-kubernetes-started-by-josh-17-4hktq-wgd0r-rm4jr | Running | Pod | 1/1 | jenkins | jessica-cluster |
| <input type="checkbox"/> | jenkins-plugin-kubernetes-started-by-josh-18-bddsw-cwmjz-bd0xw | Running | Pod | 1/1 | jenkins | jessica-cluster |
| <input type="checkbox"/> | jenkins-plugin-kubernetes-started-by-josh-19-s9l3t-cww96-ql179 | Running | Pod | 1/1 | jenkins | jessica-cluster |
| <input type="checkbox"/> | jenkins-plugin-kubernetes-started-by-josh-20-j136g-x78vb-h6t5m | Running | Pod | 1/1 | jenkins | jessica-cluster |
| <input type="checkbox"/> | jenkins-plugin-kubernetes-started-by-josh-21-fx4v3-x7v7-v4rj7 | Terminating | Pod | 1/1 | jenkins | jessica-cluster |

Finally, you can check the Jenkins interface and confirm the progress of the builds/
In my case, I have 31 builds occurring simultaneously.



One final check.

We had 31 nodes running simultaneously and all of them were terminated after the completion of the build process.

Manage nodes and clouds

Refresh status

| S | Name ↓ | Architecture | Clock Difference | Free Disk Space | Free Swap Space | Free Temp Space | Response Time |
|---|---------------|---------------|------------------|-----------------|-----------------|-----------------|---------------|
| | Built-In Node | Linux (amd64) | In sync | 15.28 GB | 0 B | 15.28 GB | 0ms |
| | Data obtained | 3 min 7 sec | 3 min 7 sec | 3 min 7 sec | 3 min 7 sec | 3 min 7 sec | 3 min 7 sec |