AWS – Connect CI/CD

Creating CI/CD Pipelines for Amazon Connect

Alex Hannah – Senior Solution Architect
Ram Srirama – Senior Partner Solution Architect
Hima Kurada – Senior Product Manager

June 2021

Agenda

- Introduction to Jenkins and CI/CD for Amazon Connect
- Jenkins Installation on EC2 / AWS
- Jenkins Basic Configuration and Plugins
- Creating your first Pipeline
- Amazon Connect CI/CD Overview
- Example 1 – Instance Configuration
- Example 2 – Contact Flow Synchronization
- Example 3 – Quick Connect Synchronization
- Example 4 – Queue Synchronization
- QA + Additional Resources

aws

Introduction to Jenkins

CI/CD for Amazon Connect

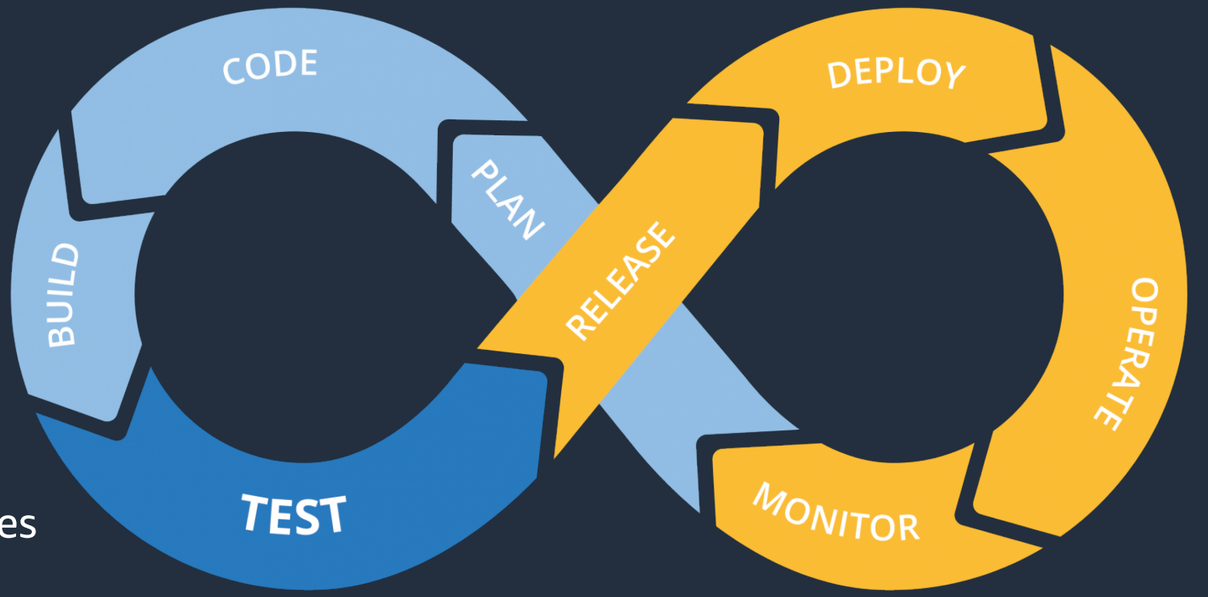
So… what is Jenkins?

- An open source automation server
- Continuous Integration and Continuous Delivery capabilities
- Jenkins is a self-contained Java based program
- Deployment options for Windows, Linux, MacOS, or UNIX
- Distributed and can run across multiple nodes or agents
- Plugins are available and custom plugins can be developed
- Scripts vs Jenkins? Why use this?

aws

What is CI/CD?

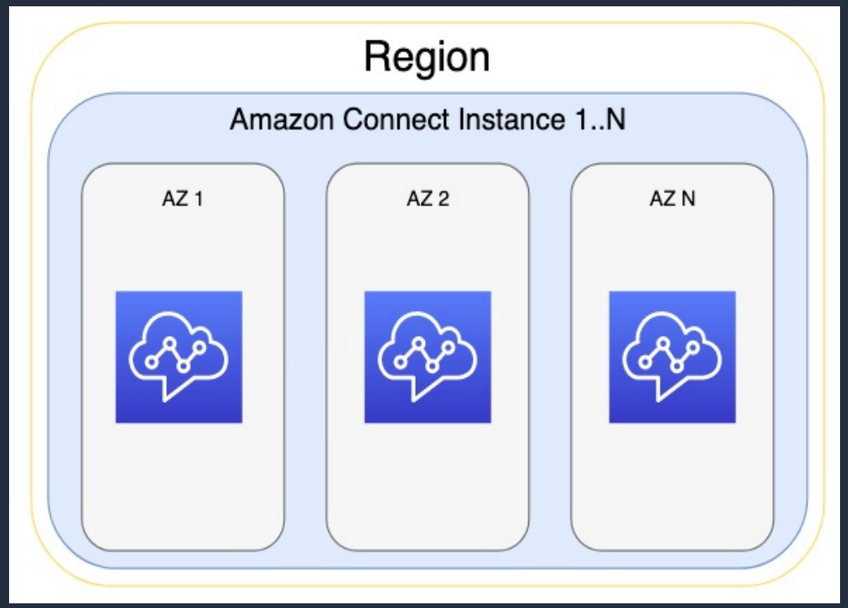
- Continuous Integration (CI) and Continuous Deliver (CD)
- Principals and Practices for delivering code changes
- Allows for frequent and reliable code deployments
- Part of DevOps and Agile software development methodologies
- CI makes use of small code changes checked into Source Code Management (SCM) such as Git or GitHub
- SCM allows for Branching
- CD picks up where CI leaves off, this creates automated ways to deliver applications/code to a selected infrastructure or environment (think Connect Instance).



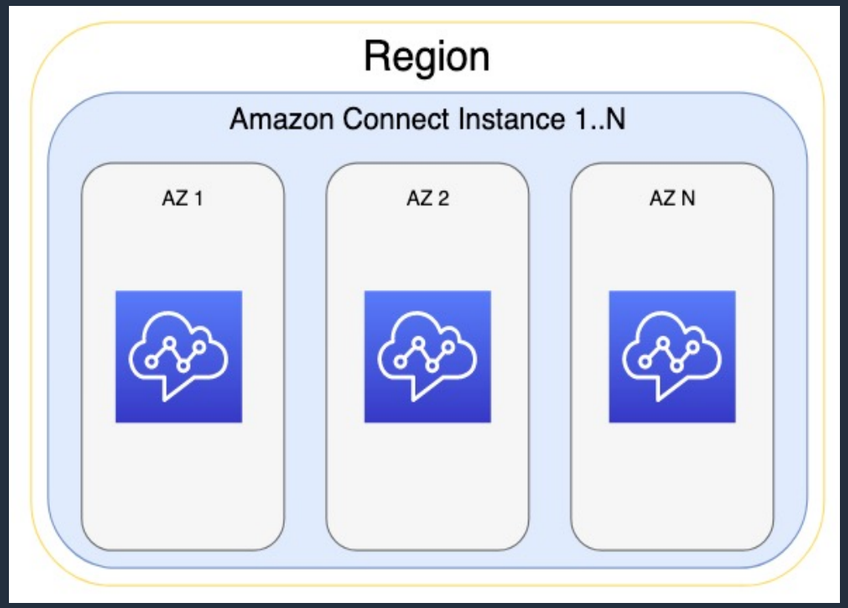
aws

Amazon Connect Instance Synchronization

Primary Instance – US-EAST-1



Secondary Instance – US-WEST-2

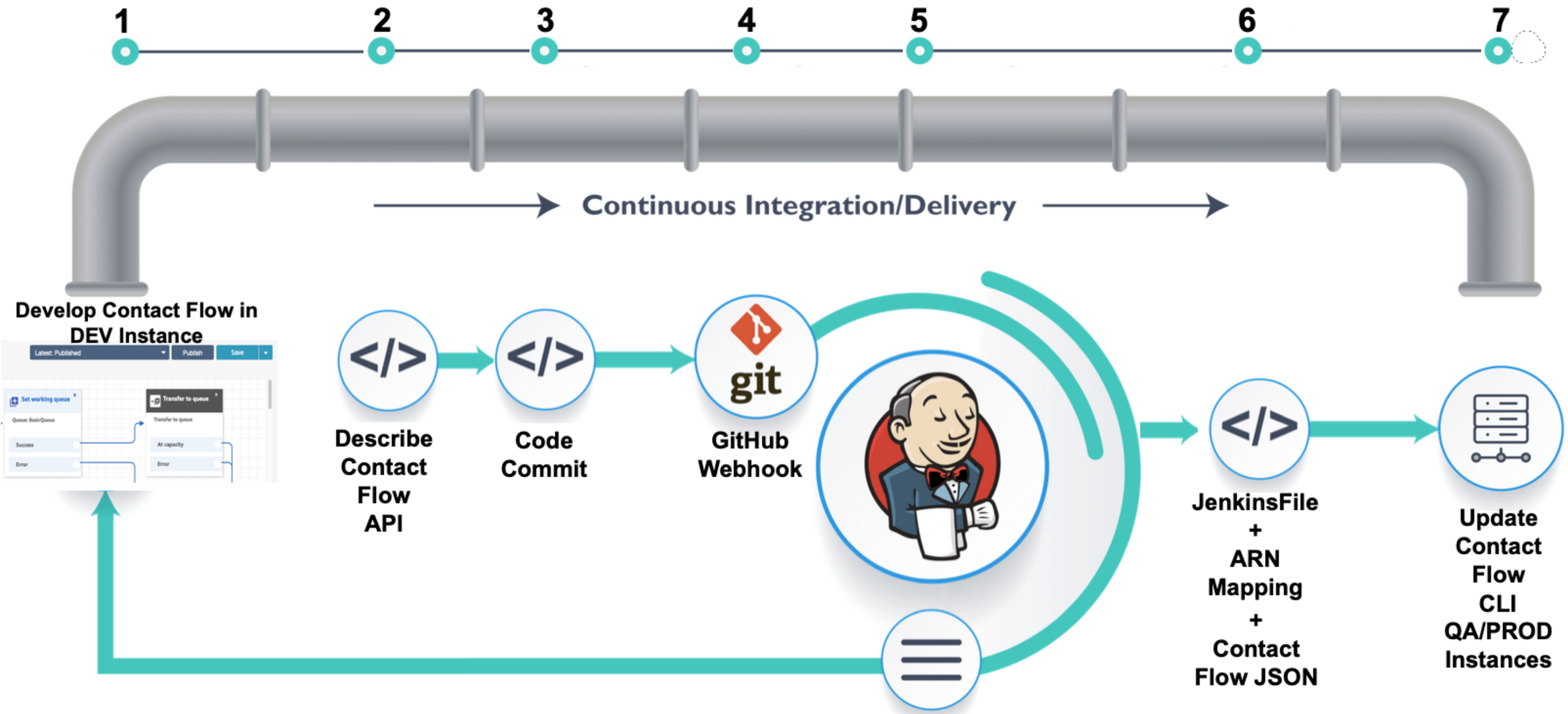


Question: How do I keep my instance configurations and changes synchronized between a Dev, Staging, and Production?

Question: How do I keep my instance configurations and changes synchronized between regional failover instances?

aws

Amazon Connect CI/CD Workflow



aws

Installing Jenkins

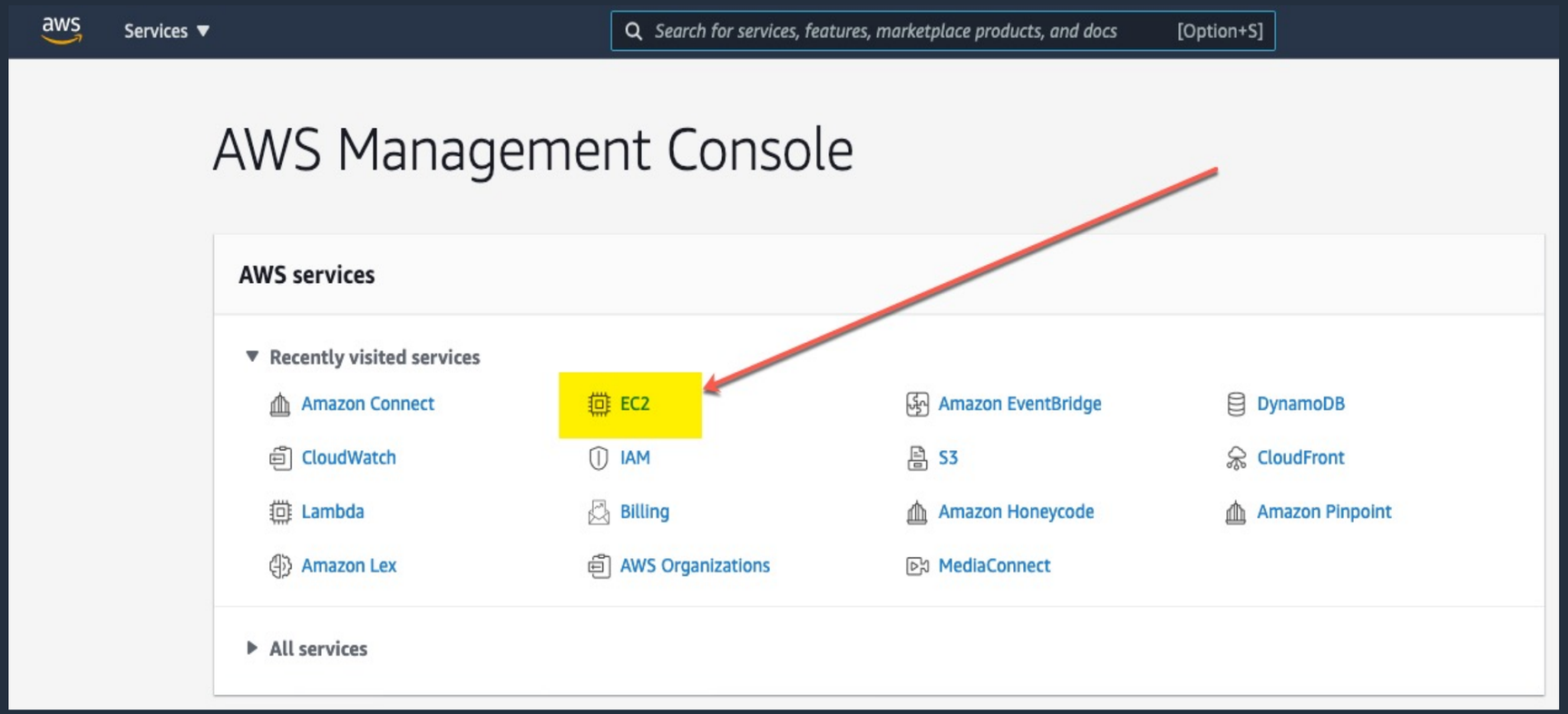
Configuration for an AWS EC2 Instance

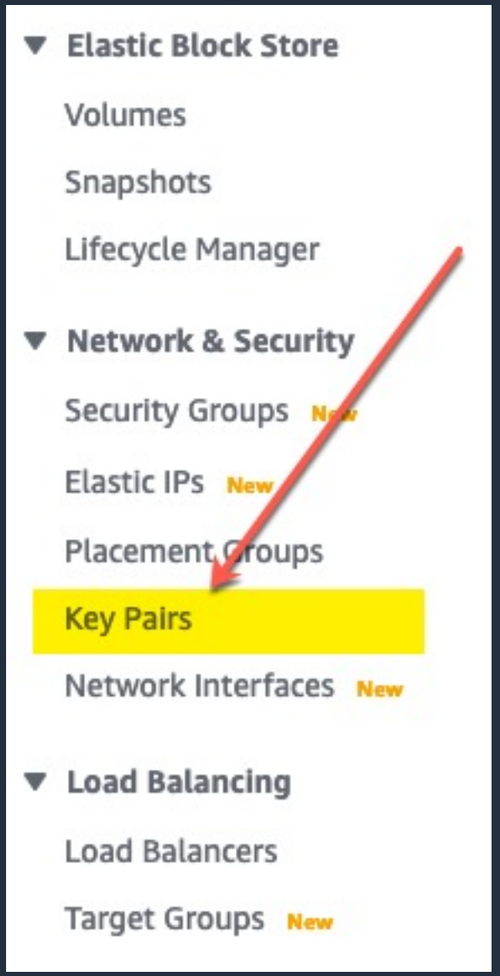
Installing Jenkins on AWS EC2

- Create a security key pair for SSH Access to EC2 Linux Instance
- Create a security group with port exceptions for traffic
- Launch the EC2 Instance
- Map the Security Group
- Map the Security Keys
- Install Jenkins on the Instance
- Configure Jenkins

aws

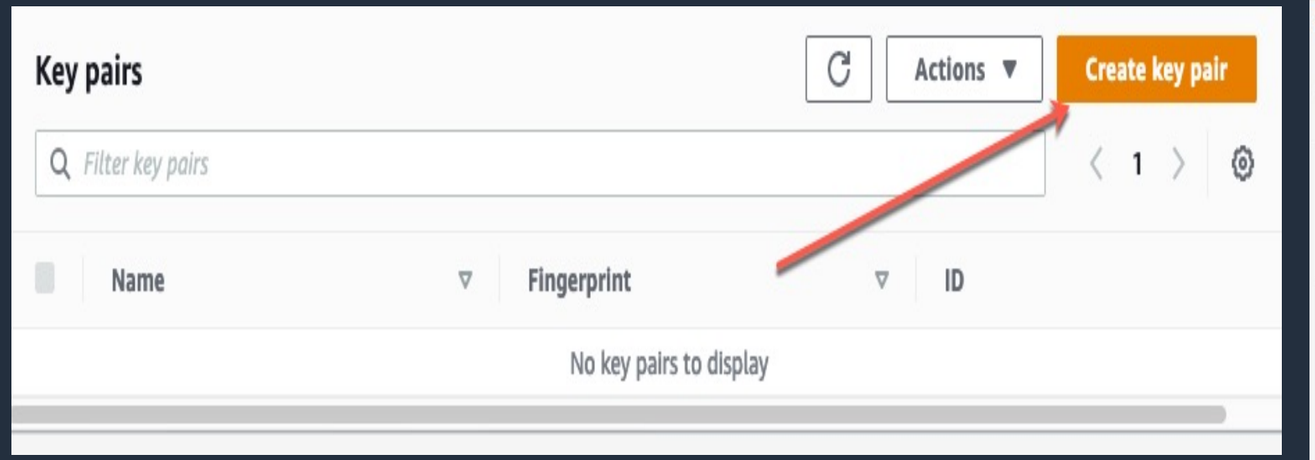
Step 1: Security Key Pairs



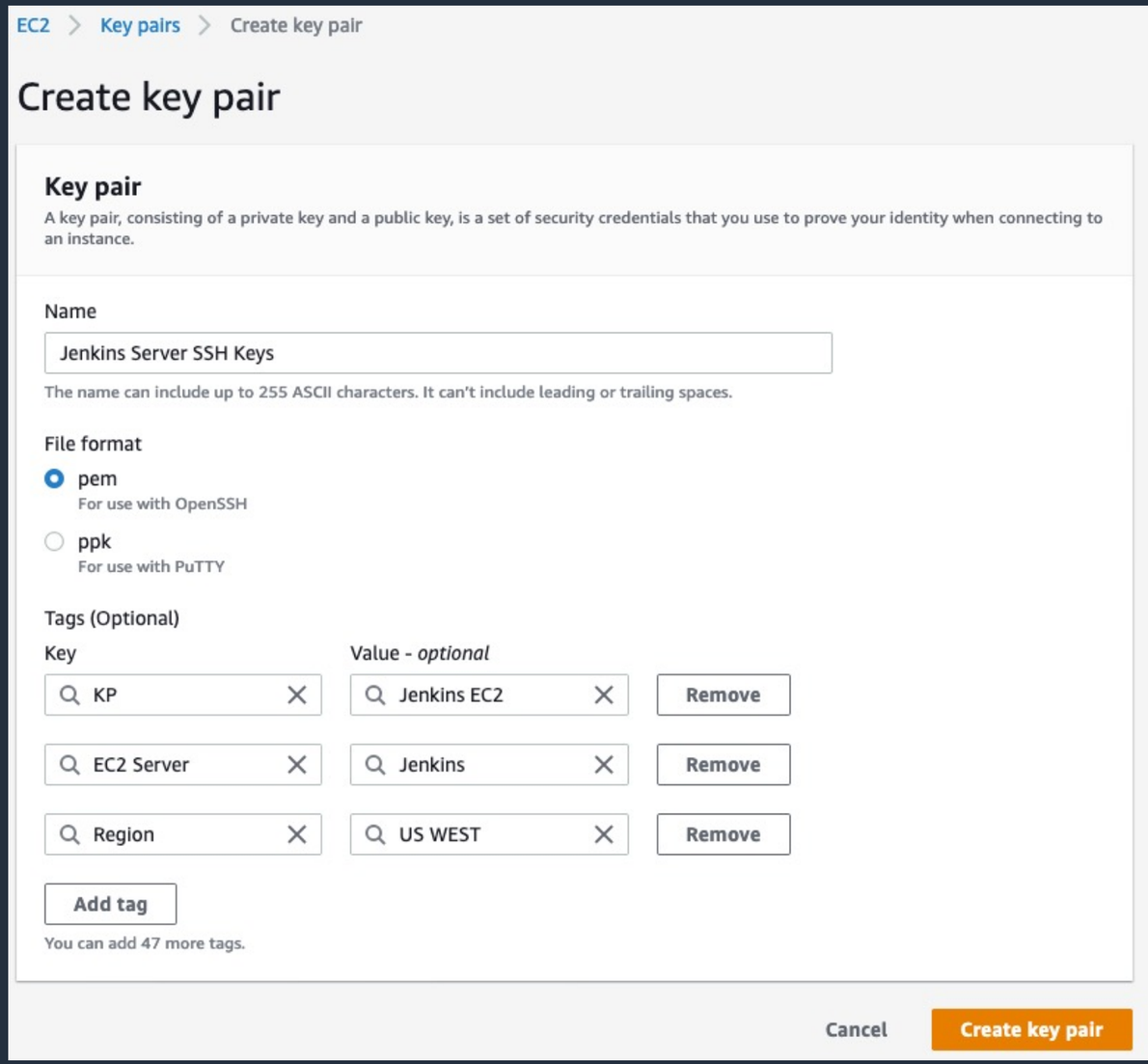


- Configuration of the Security Key Pairs (SSH Keys) will be done under EC2
- https://console.aws.amazon.com/ec2/
- Security Keys Pairs are under the Network & Security menu on the left hand side

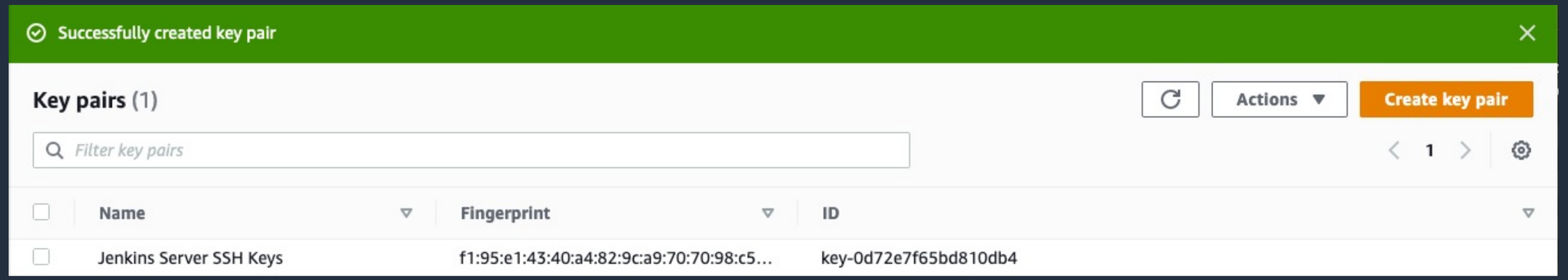
Creating the Key Pairs



- Key Pairs are essentially SSH credentials to log into your EC2 Instance. It is critical to keep these secure, anyone with access will be able to log into your Jenkins server, RBAC best practices should apply here!
- PEM Format works well for most SSH Clients (Mac/Windows)
- Tags are optional but helpful!



Saving the Key Pairs

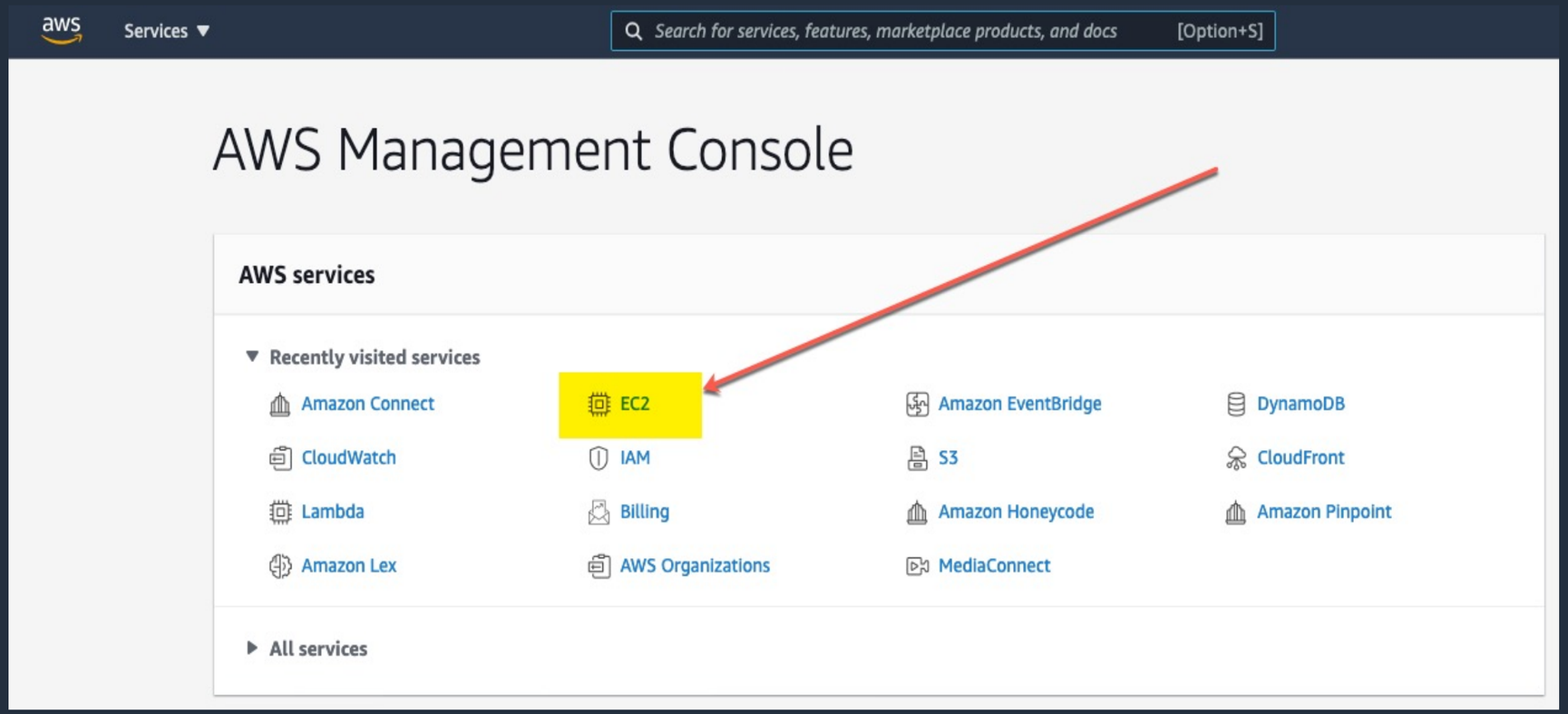


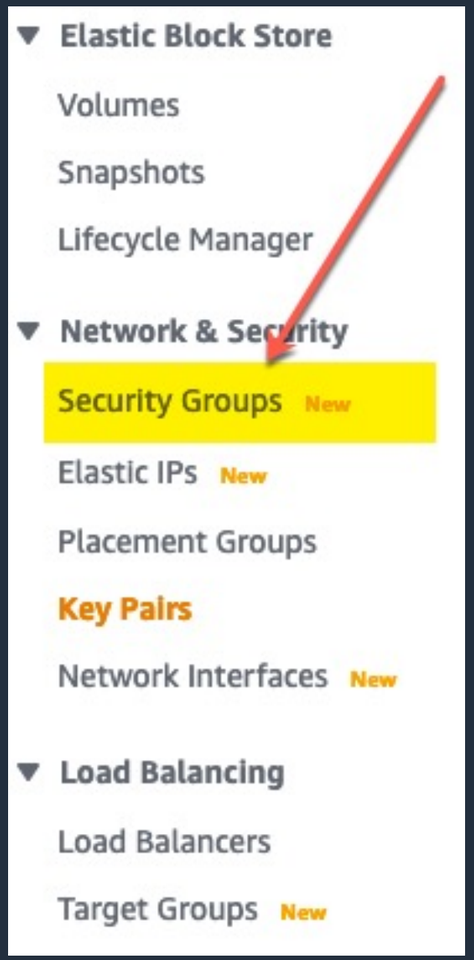
- Save the Key Pairs somewhere secure, grant access only to those who need it!
- Download and store the PEM Format SSH Keys
- CHMOD the file on your local machine to lock down access!

```
wllhann@38f9d3c82d1d Jenkins PEM File % ls
Jenkins Server SSH Keys.pem
wllhann@38f9d3c82d1d Jenkins PEM File %
wllhann@38f9d3c82d1d Jenkins PEM File %
wllhann@38f9d3c82d1d Jenkins PEM File % chmod 400 JenkinsServer-SSHKeys.pem
wllhann@38f9d3c82d1d Jenkins PEM File %
wllhann@38f9d3c82d1d Jenkins PEM File % ls -l JenkinsServer-SSHKeys.pem
-r--------@ 1 wllhann  staff  1674 Jun 10 13:25 JenkinsServer-SSHKeys.pem
wllhann@38f9d3c82d1d Jenkins PEM File %
wllhann@38f9d3c82d1d Jenkins PEM File %
wllhann@38f9d3c82d1d Jenkins PEM File %
```

aws

Step 2: Security Groups

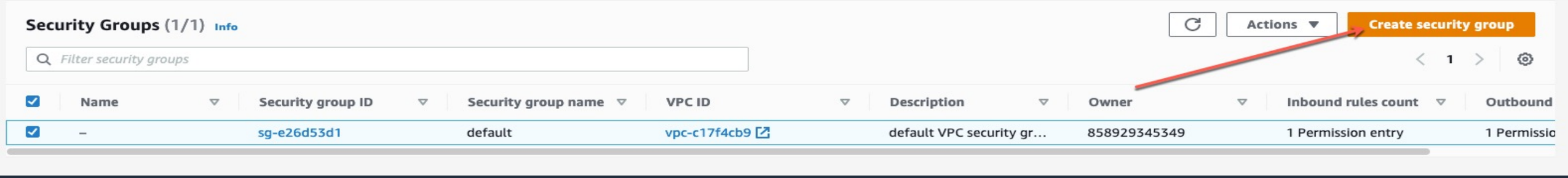


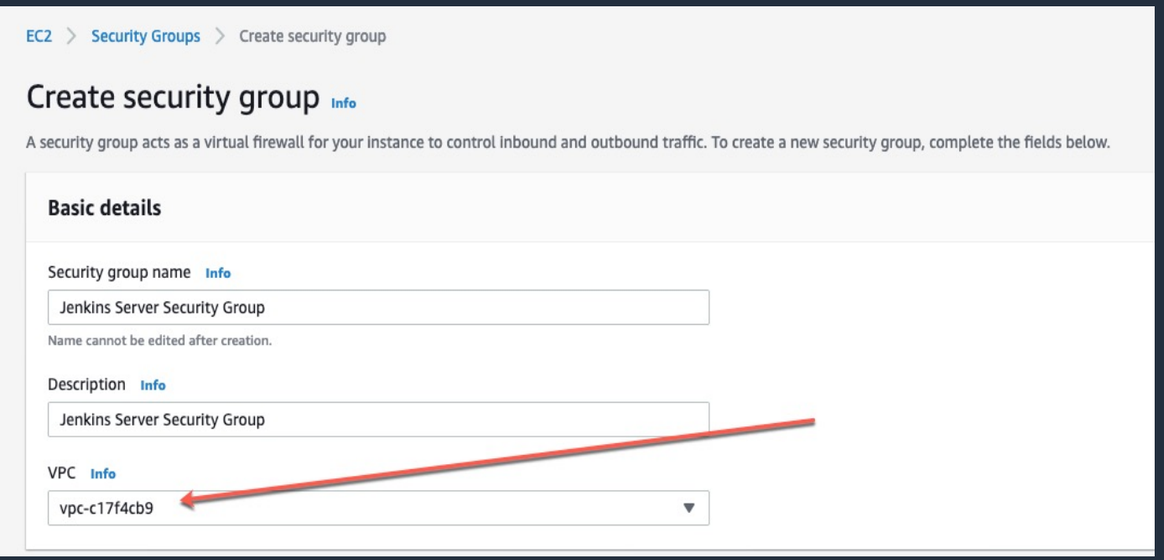


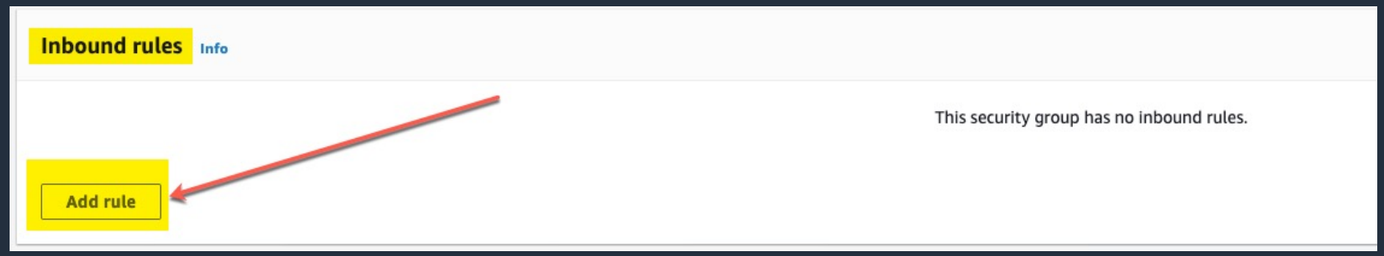
- Configuration of the Security Groups will be done under EC2
- https://console.aws.amazon.com/ec2/
- Security Groups are under the Network & Security menu on the left hand side

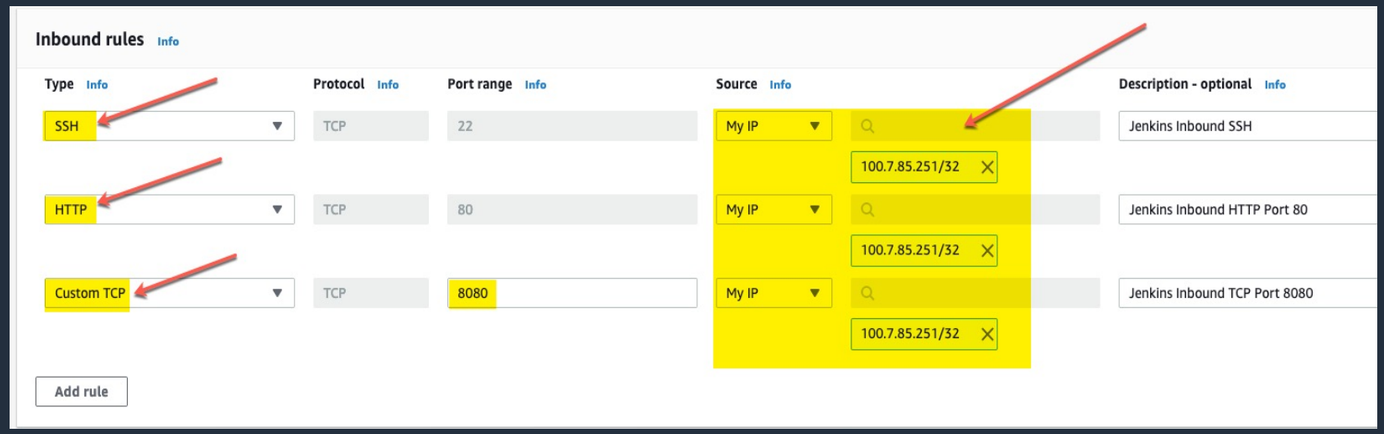
aws

Creating the Security Group









- Plan out your Security Group Name
- VPC Configuration is a pre-req!
- Inbound Security Rules will be created
- Lock down your box!!!

aws

Verify the Security Group Settings!!!

EC2 > Security Groups > sg-095b8df85e4b793c7 - Jenkins Server Security Group

sg-095b8df85e4b793c7 - Jenkins Server Security Group

Actions

Details

Security group name	Security group ID	Description	VPC ID
Jenkins Server Security Group	sg-095b8df85e4b793c7	Jenkins Server Security Group	vpc-c17f4cb9

Owner	Inbound rules count	Outbound rules count
858929345349	3 Permission entries	1 Permission entry

Inbound rules | Outbound rules | Tags

Inbound rules (3)

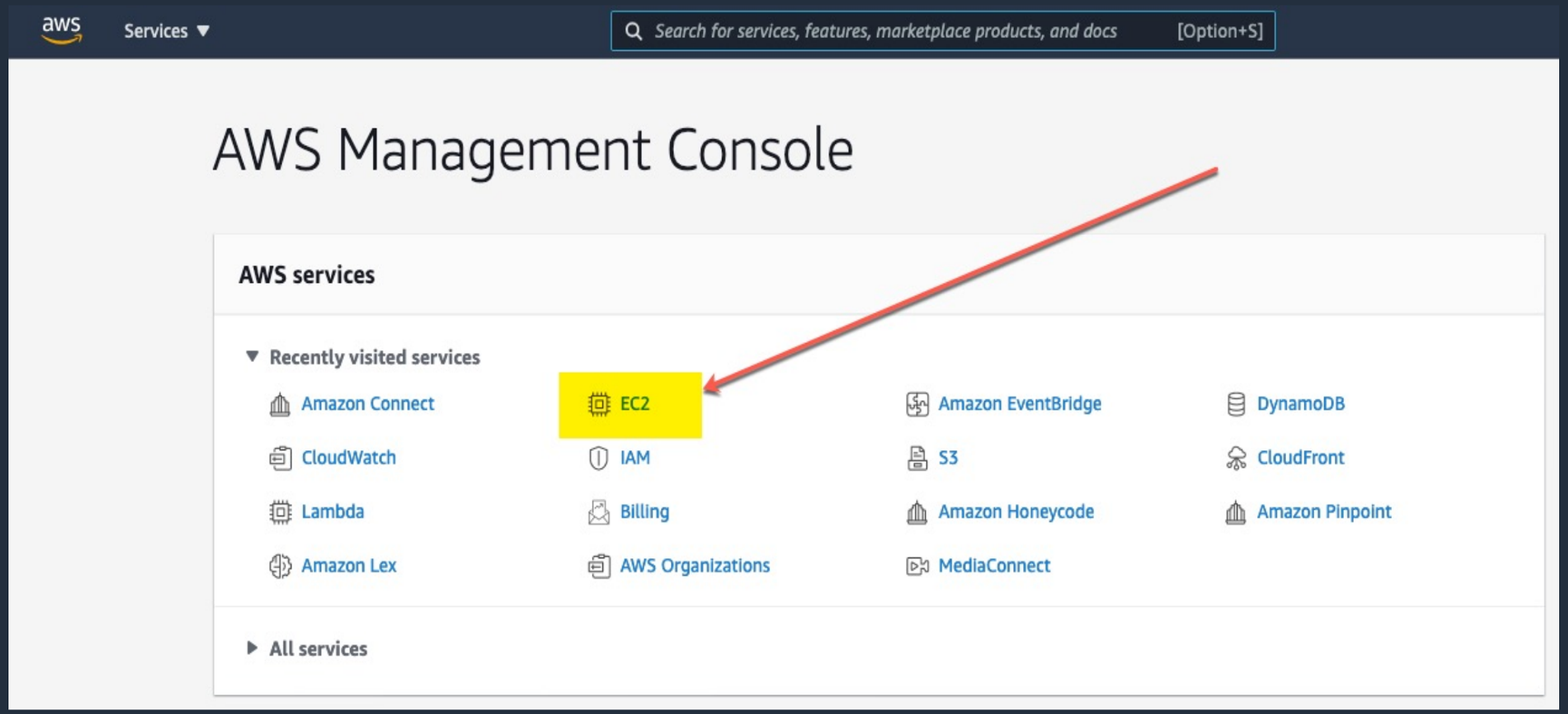
Edit inbound rules

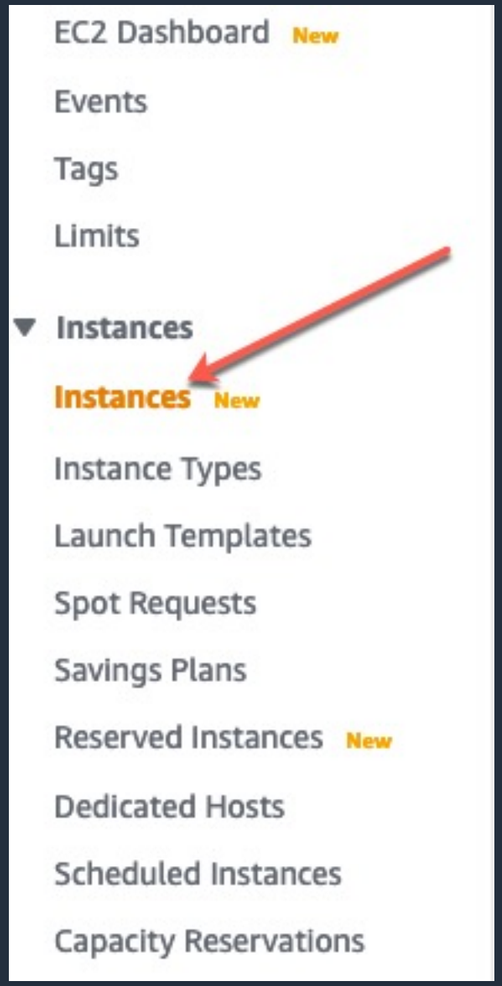
Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	100.7.85.251/32	Jenkins Inbound HTTP Port 80
Custom TCP	TCP	8080	100.7.85.251/32	Jenkins Inbound TCP Port 8080
SSH	TCP	22	100.7.85.251/32	Jenkins Inbound SSH

- Double check your Security Group configuration to ensure you have the right IP Addresses and Ports allowed!

aws

Step 3: Launch the Instance

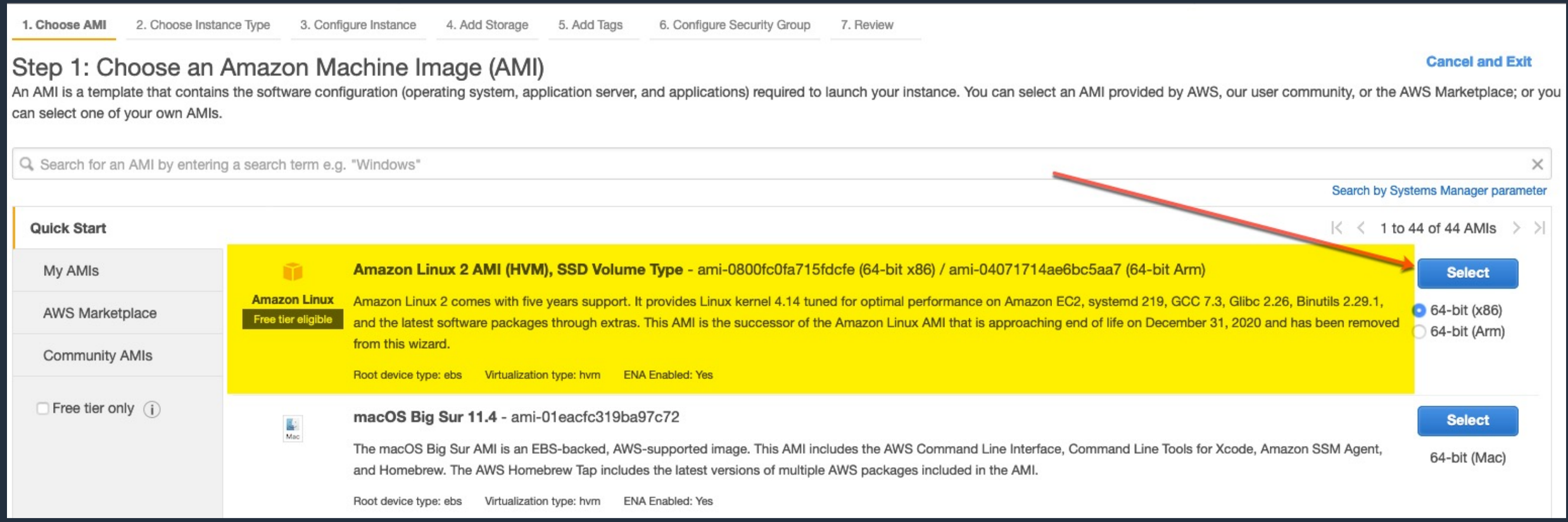




- Configuration of the Instances will be done under EC2
- https://console.aws.amazon.com/ec2/
- Instances are under the Instances menu on the left hand side

Creating the Instance

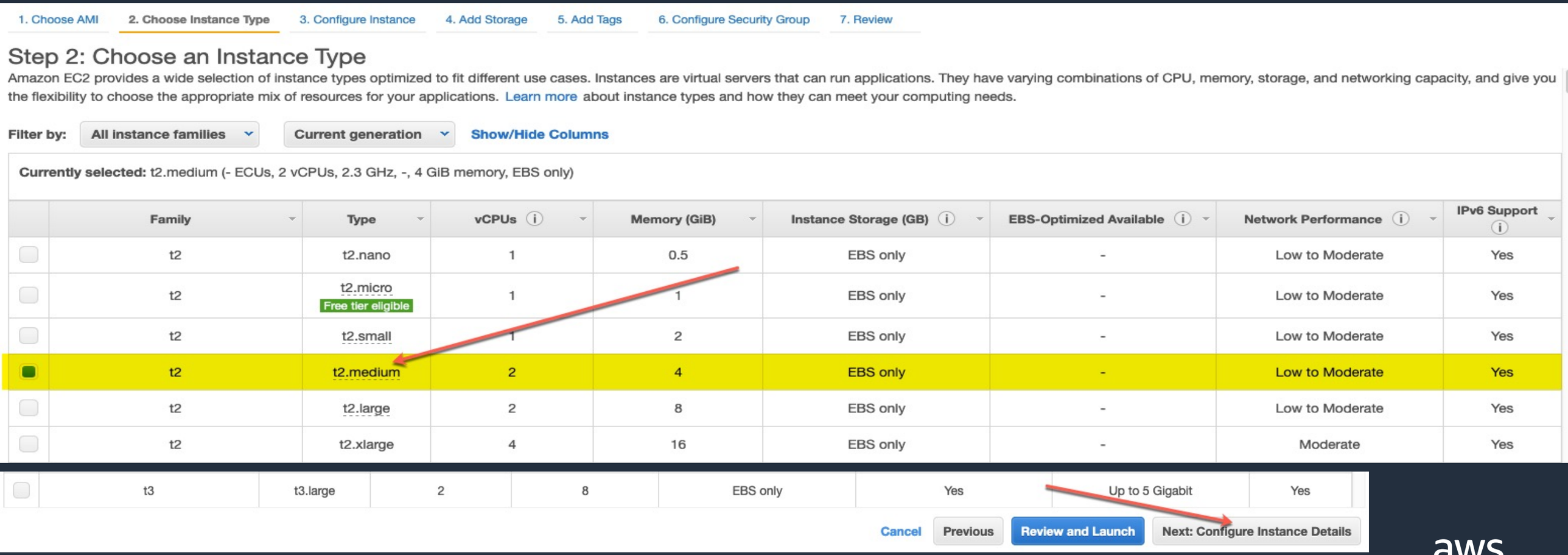
- Instance Type: Amazon Linux 2 AMI (HVM), SSD Volume Type
- 64-bit (x86)



Creating the Instance

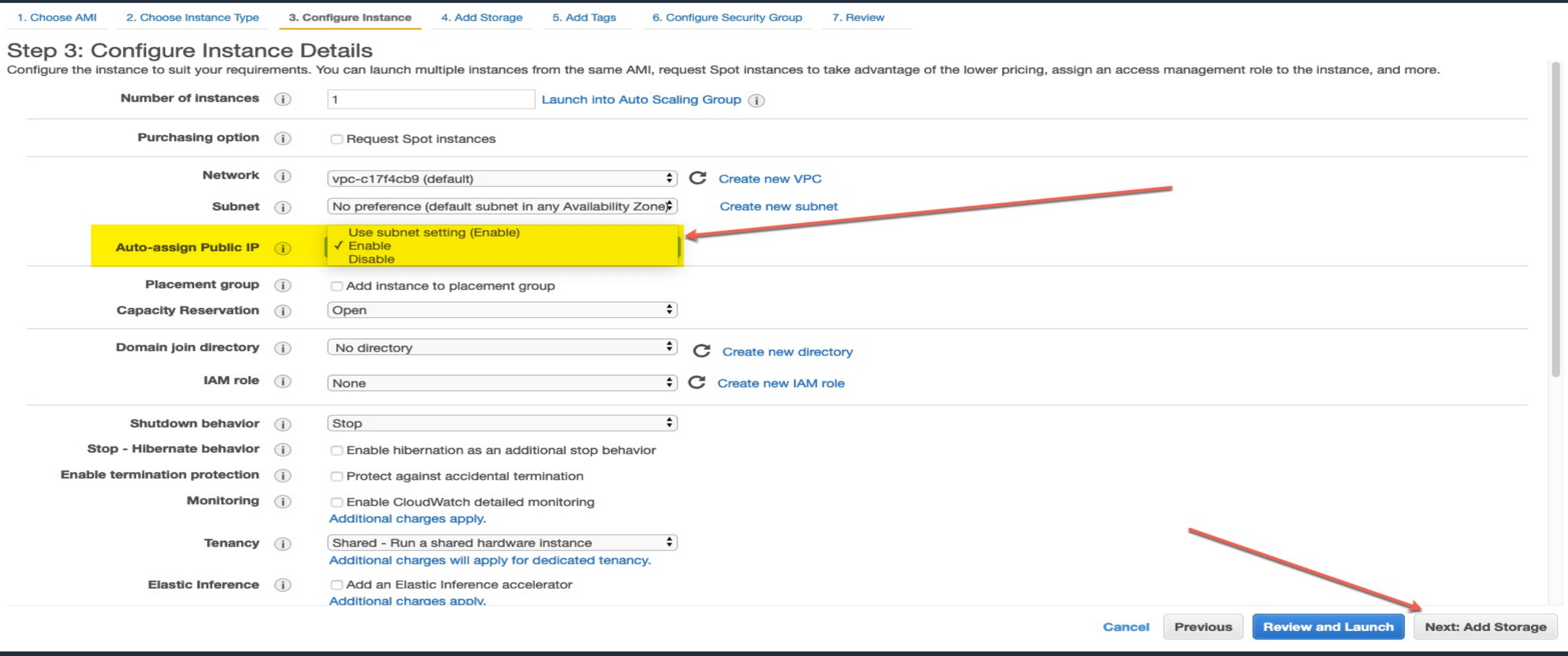
- Instance Type: t2.medium
- 2 vCPU's
- 4 GiB Memory

Note: During our testing this instance type provided decent speed/performance, you may need to increase resources accordingly based on your job complexity and needs!!!

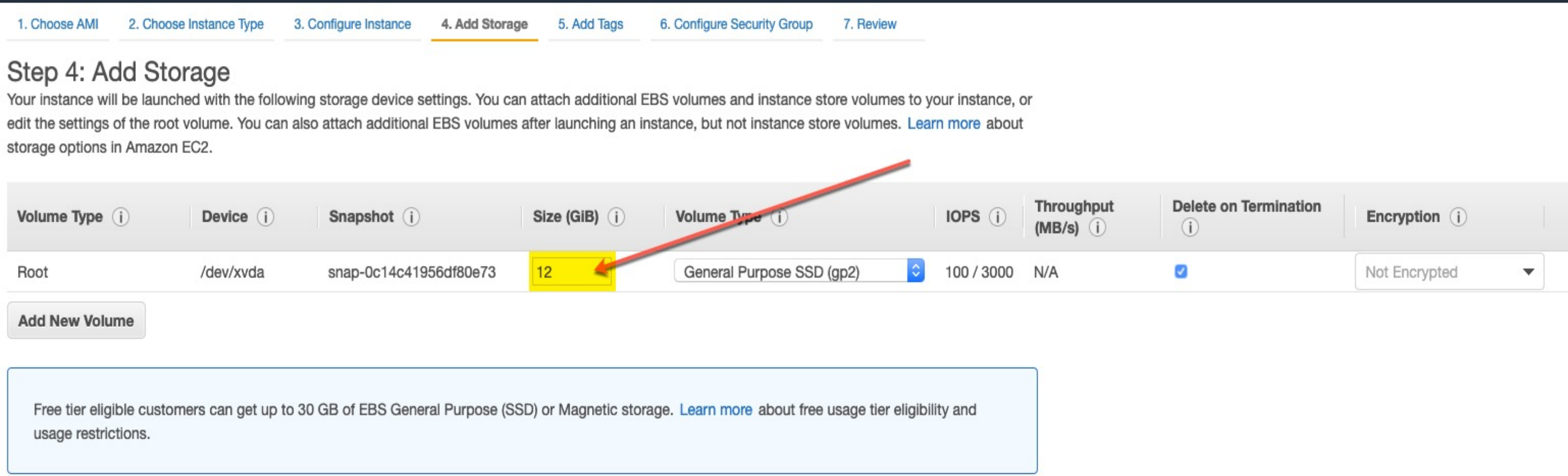


Assigning Instance Details – Public IP

- A Public IP or NAT using a Public IP will be required for GitHub to communicate properly with Jenkins



Assigning Instance Details – HDD Size

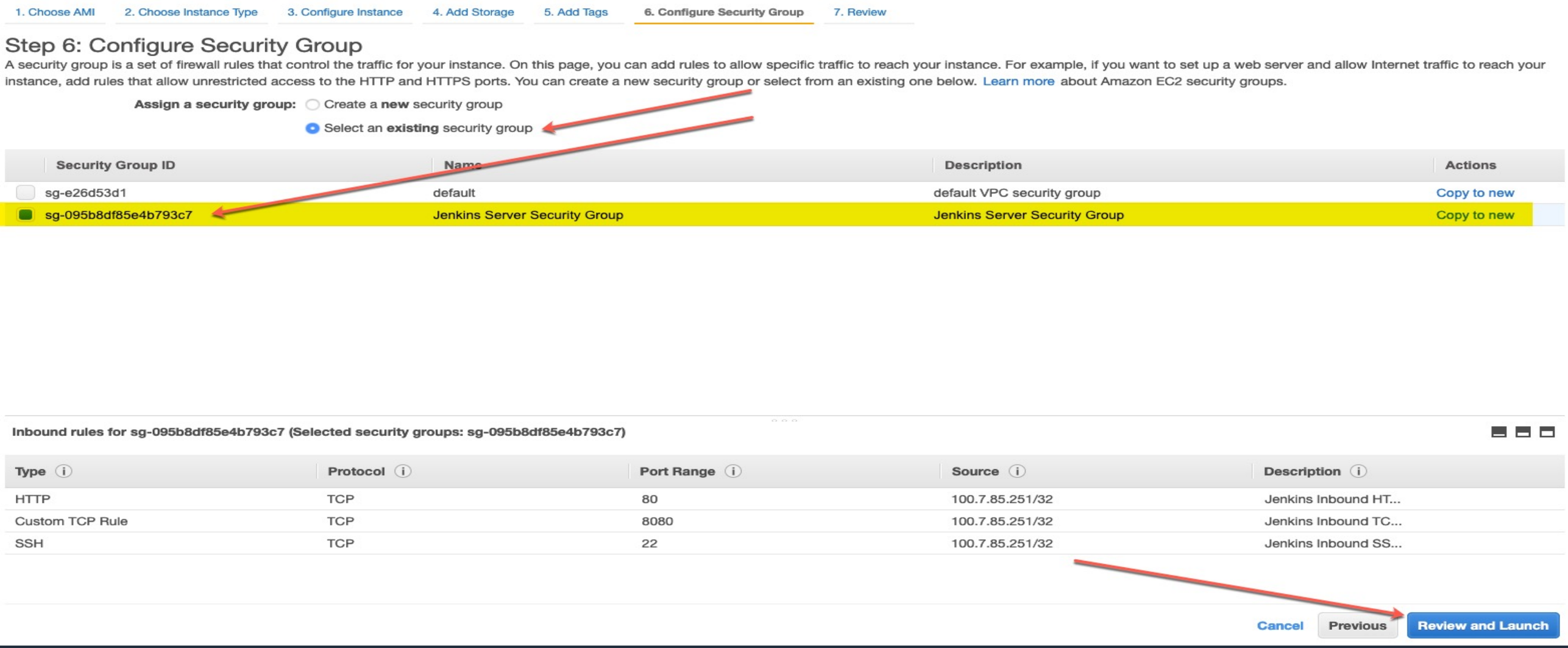


- For our demo instance, we are increasing the SSD HDD size to 12 GiB
- Jobs in Jenkins are all XML files called Jenkins Files and hosted in GitHub
- Minimal HDD size is necessary to run Jenkins for Amazon Connect CI/CD, most configurations are in GitHub!

Note: If you are using Jenkins for local software compilation, this settings will likely be increased for Maven builds, etc. For our demo, we are strictly using Github, JSON/XML, and a minimal configuration of Jenkins.

aws

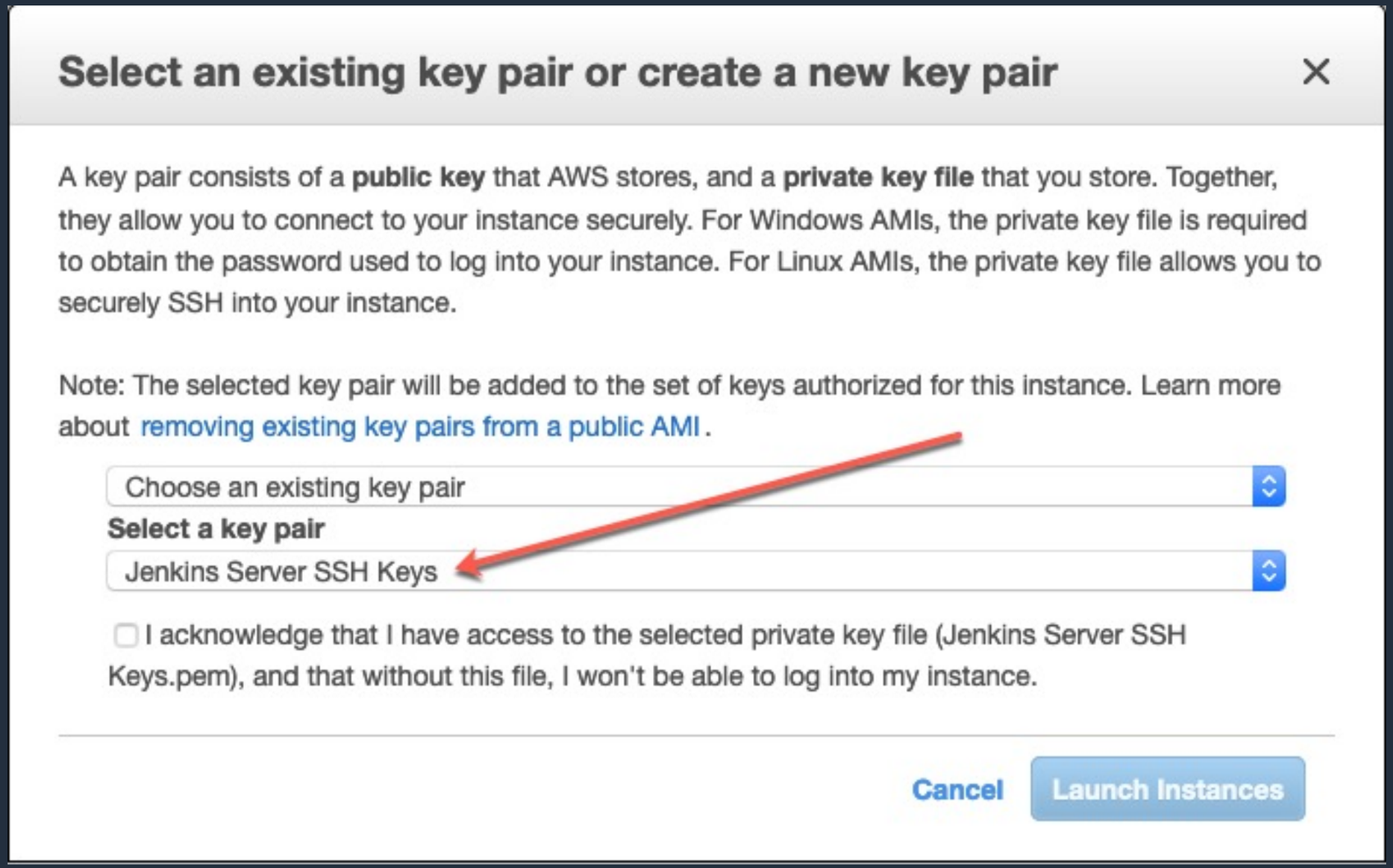
Assigning Instance Details – Security Group



- For our demo instance, we will assign the Security Group we previously configured.

aws

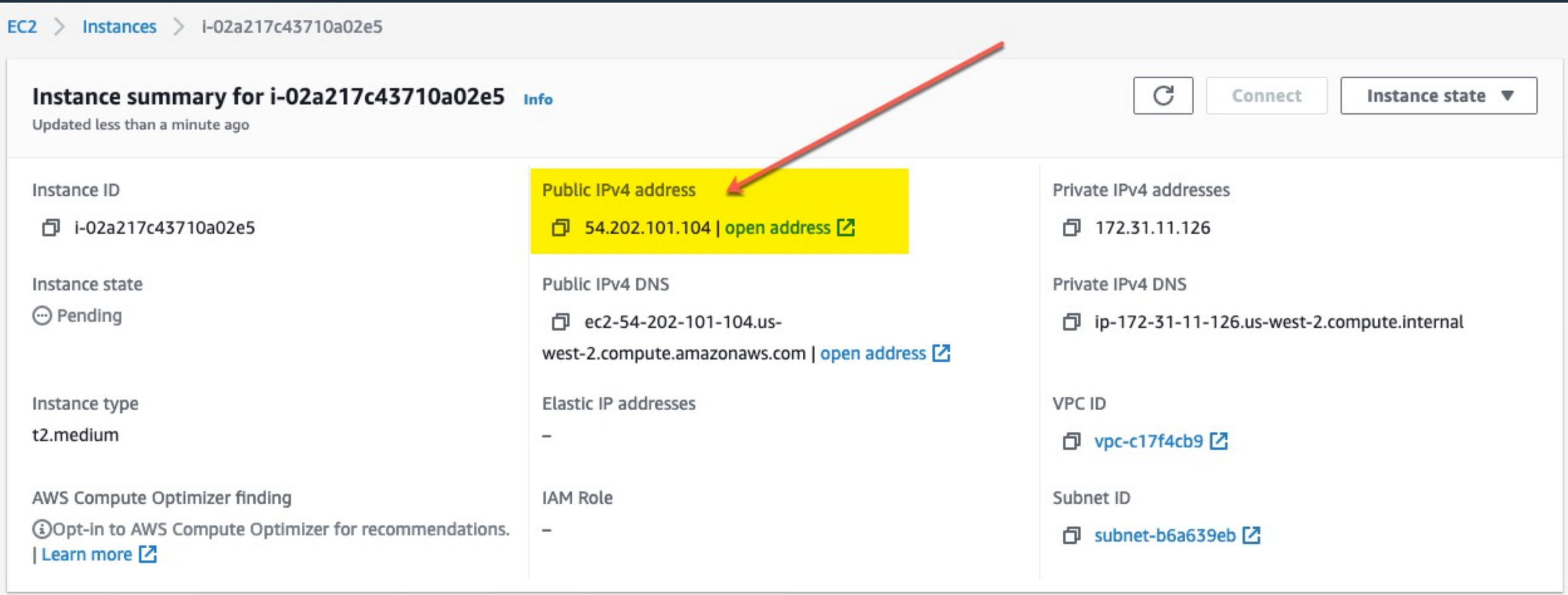
Assigning Instance Details – Assign a Security Key Pair



- For our demo instance, we will assign the Security Key Pair we previously configured.

aws

Determining your Public IP



- Once your instance is configured, and you launch it, we need to determine our Public IP Address. Please note this IP, you will be using it for SSH Access and Jenkins installation.

aws

SSH into the EC2 Instance and Update!

```
[wllhann@38f9d3c82d1d desktop % cd 'Jenkins CICD Preso'
[wllhann@38f9d3c82d1d Jenkins CICD Preso % ls
 AWS - Connect - Jenkins CI:CD Preso v1.pptx     Jenkins PEM File
 AWS - Connect - Jenkins CICD Preso v1.pptx      ~$AWS - Connect - Jenkins CICD Preso v1.pptx
[wllhann@38f9d3c82d1d Jenkins CICD Preso % cd 'Jenkins PEM File'
[wllhann@38f9d3c82d1d Jenkins PEM File % ls
 JenkinsServer-SSHKeys.pem
[wllhann@38f9d3c82d1d Jenkins PEM File %
[wllhann@38f9d3c82d1d Jenkins PEM File %
[wllhann@38f9d3c82d1d Jenkins PEM File %
[wllhann@38f9d3c82d1d Jenkins PEM File % ssh -i JenkinsServer-SSHKeys.pem ec2-user@54.202.101.104
The authenticity of host '54.202.101.104 (54.202.101.104)' can't be established.
ECDSA key fingerprint is SHA256:dJoYGZcRPqbKxGtp0EqaUjiruAgo0diBUZsvFkkSB/A.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.202.101.104' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-11-126 ~]$
```

- SSH Command on MAC:
  ssh –i {your PEM file} ec2user@{public IP}

```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[[ec2-user@ip-172-31-11-126 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
No packages marked for update
[ec2-user@ip-172-31-11-126 ~]$
```

- YUM Update Command:
  sudo yum update -y

aws

Install Jenkins on your EC2 Instance

```
[ec2-user@ip-172-31-11-126 ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \
>     https://pkg.jenkins.io/redhat-stable/jenkins.repo
--2021-06-10 19:02:48--  https://pkg.jenkins.io/redhat-stable/jenkins.repo
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.54.133, 2a04:4e42:d::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.54.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 85
Saving to: '/etc/yum.repos.d/jenkins.repo'

100%[======================================================================================================================================>] 85          --.-K/s   in 0s

2021-06-10 19:02:49 (4.86 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [85/85]

[ec2-user@ip-172-31-11-126 ~]$
```

```
[ec2-user@ip-172-31-11-126 ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
[ec2-user@ip-172-31-11-126 ~]$
```

```
[ec2-user@ip-172-31-11-126 ~]$ sudo yum upgrade
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                               | 3.7 kB  00:00:00
jenkins                                                                  | 2.9 kB  00:00:00
jenkins/primary_db                                                       |  37 kB  00:00:00
No packages marked for update
[ec2-user@ip-172-31-11-126 ~]$
```

```
[ec2-user@ip-172-31-11-126 ~]$
[ec2-user@ip-172-31-11-126 ~]$ sudo yum install jenkins java-1.8.0-openjdk-devel -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
```

```
[ec2-user@ip-172-31-11-126 ~]$
[ec2-user@ip-172-31-11-126 ~]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-11-126 ~]$
[ec2-user@ip-172-31-11-126 ~]$ sudo systemctl start jenkins
[ec2-user@ip-172-31-11-126 ~]$
[ec2-user@ip-172-31-11-126 ~]$ sudo systemctl status jenkins
● jenkins.service - LSB: Jenkins Automation Server
   Loaded: loaded (/etc/rc.d/init.d/jenkins; bad; vendor preset: disabled)
   Active: active (running) since Thu 2021-06-10 19:09:23 UTC; 9s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 3832 ExecStart=/etc/rc.d/init.d/jenkins start (code=exited, status=0/SUCCESS)
```


aws

Update AWS CLI on your EC2 Instance

- Update the AWS CLI on the Jenkins EC2 Instance
- Default version is 1.18.x , upgrade to 2.2.11

```
[ec2-user@ip-172-31-11-126 /]$ sudo rm -rf /bin/aws
```

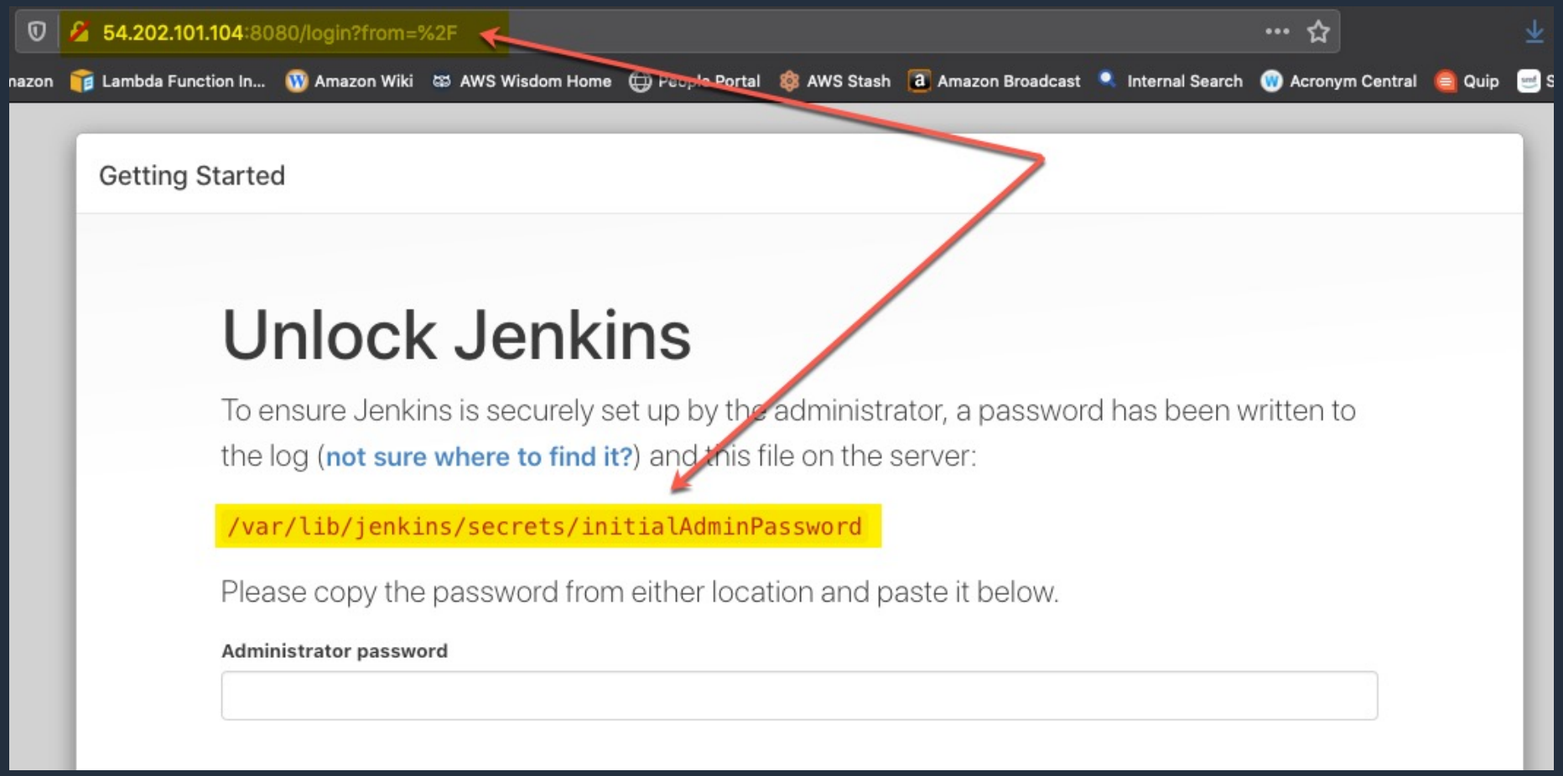
```
[[ec2-user@ip-172-31-11-126 ~]$
[[ec2-user@ip-172-31-11-126 ~]$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 41.6M  100 41.6M    0     0  97.1M      0 --:--:-- --:--:-- --:--:-- 97.1M
[[ec2-user@ip-172-31-11-126 ~]$
[[ec2-user@ip-172-31-11-126 ~]$
[[ec2-user@ip-172-31-11-126 ~]$ ls
awscliv2.zip
[[ec2-user@ip-172-31-11-126 ~]$
[[ec2-user@ip-172-31-11-126 ~]$
[[ec2-user@ip-172-31-11-126 ~]$ unzip awscliv2.zip
Archive:  awscliv2.zip
   creating: aws/
   creating: aws/dist/
  inflating: aws/THIRD_PARTY_LICENSES
  inflating: aws/install
  inflating: aws/README.md
   creating: aws/dist/_struct/
```

```
[[ec2-user@ip-172-31-11-126 ~]$
[[ec2-user@ip-172-31-11-126 ~]$ sudo ./aws/install -i /usr/local/aws -b /bin
You can now run: /bin/aws --version
[[ec2-user@ip-172-31-11-126 ~]$
[[ec2-user@ip-172-31-11-126 ~]$
[[ec2-user@ip-172-31-11-126 ~]$
[[ec2-user@ip-172-31-11-126 ~]$ aws --version
aws-cli/2.2.11 Python/3.8.8 Linux/4.14.232-176.381.amzn2.x86_64 exe/x86_64.amzn.2 prompt/off
[[ec2-user@ip-172-31-11-126 ~]$
```


aws

Browse to Jenkins and initial login

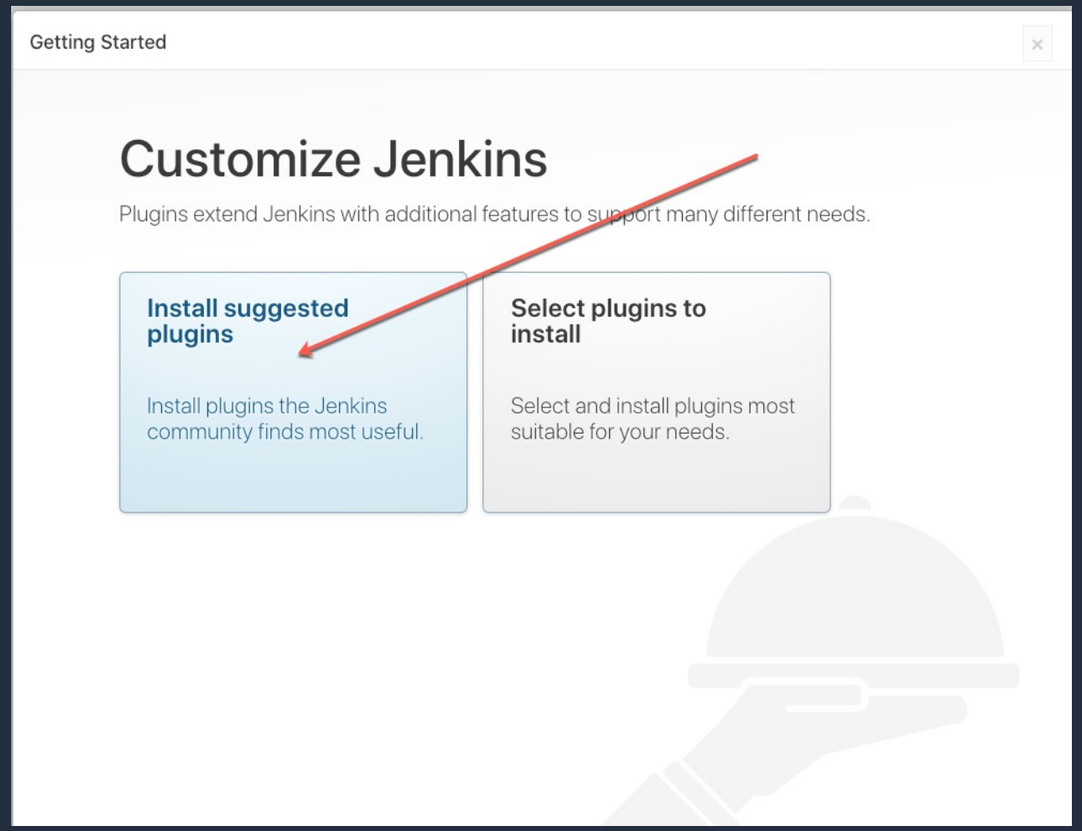
- Initial Jenkins Password is stored in a file on the OS.
- It is used for first time login only and changed when you configure the first admin account!
- Can use the CAT command to view it on the CLI

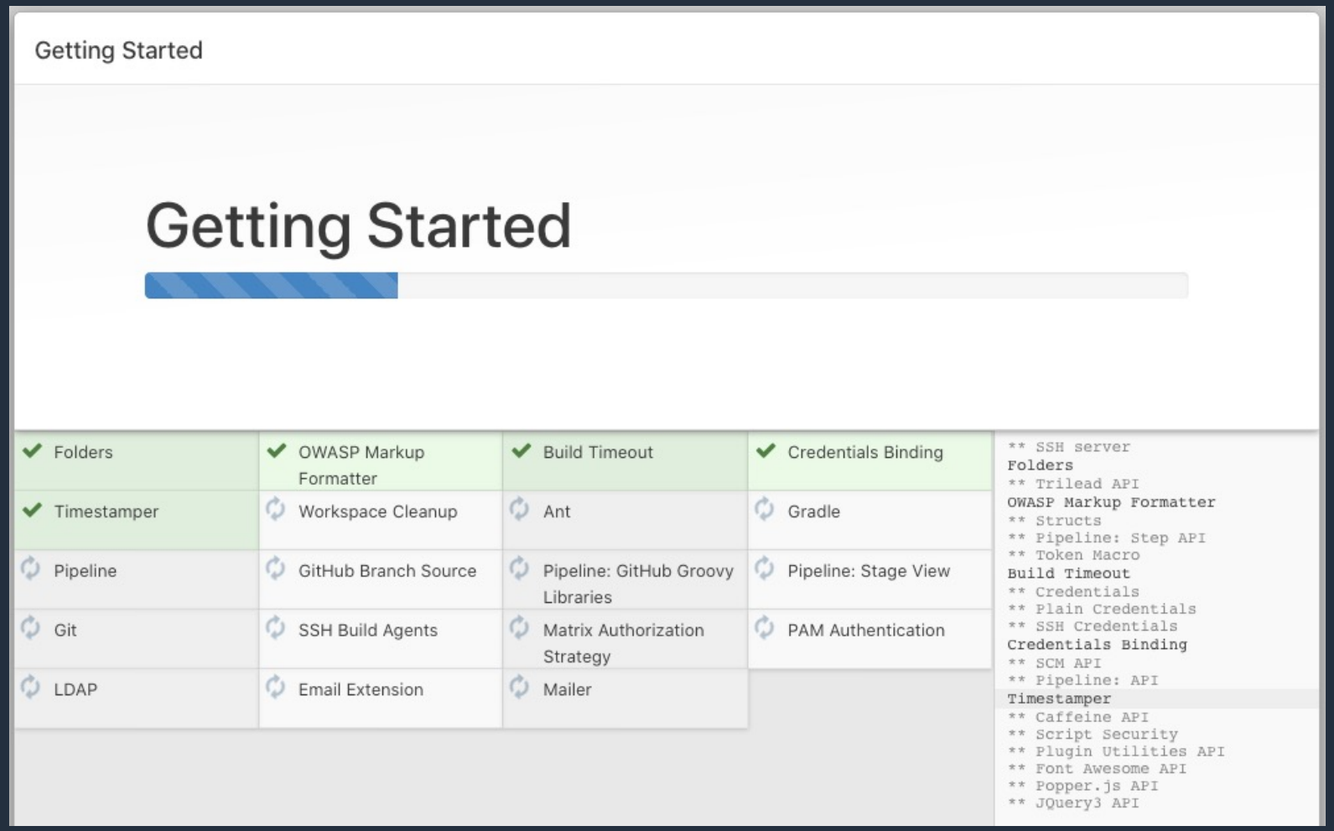



```
[ec2-user@ip-172-31-11-126 ~]$
[ec2-user@ip-172-31-11-126 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
cb523d1315654236b6ca0f80e0a92010
[ec2-user@ip-172-31-11-126 ~]$
```


aws

Setup Suggested Plugins

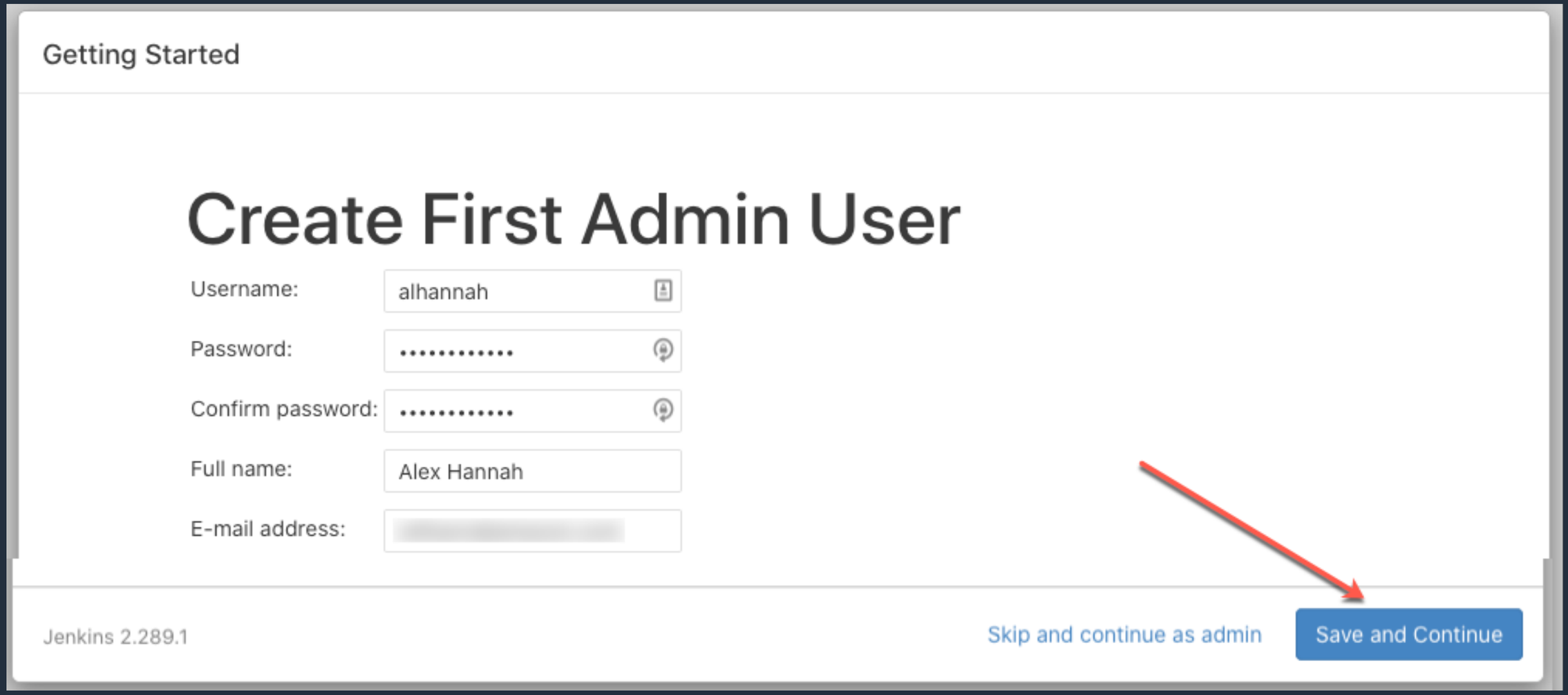




- For our demo, we will install the suggested plugins
- This will take 3-5 minutes to complete, be patient!

aws

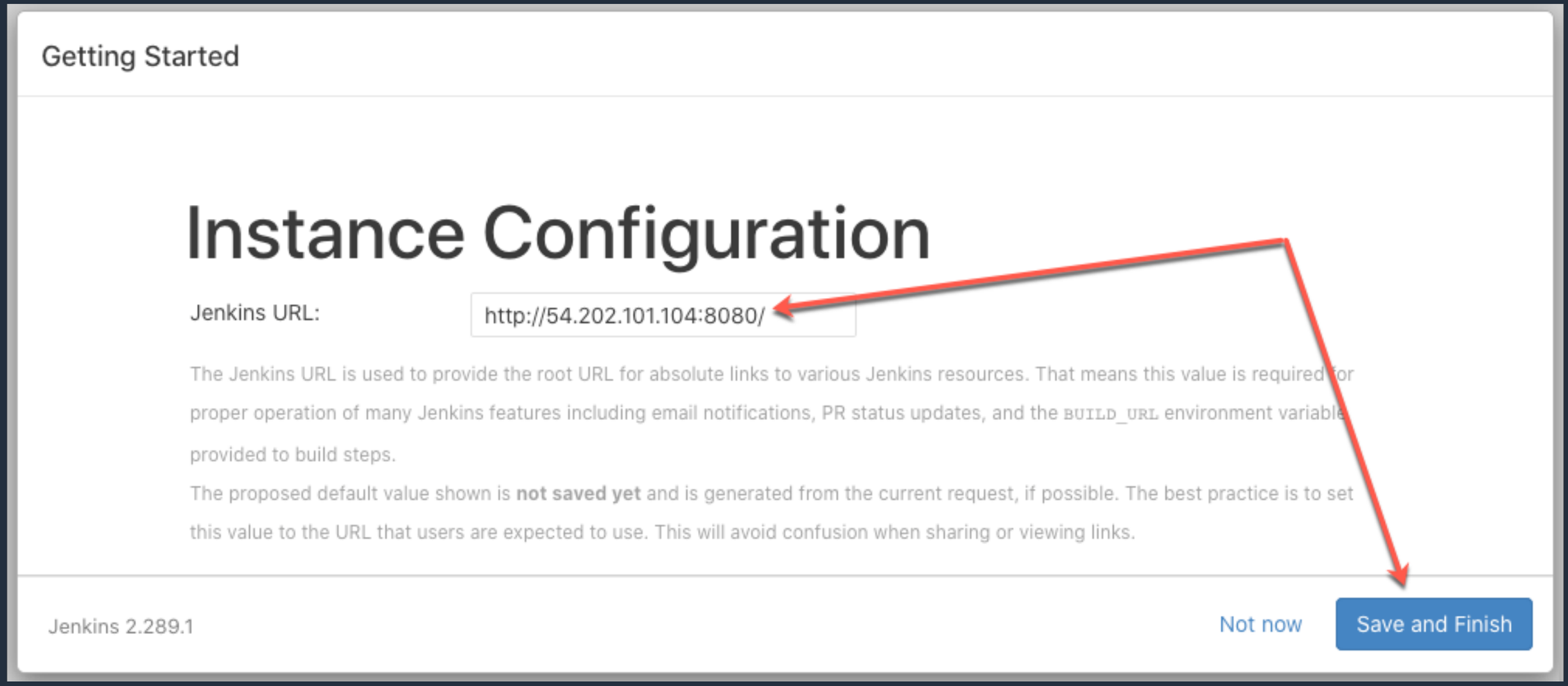
Create the Admin User Account



- Additional Administrator accounts can be configured after setup is completed

aws

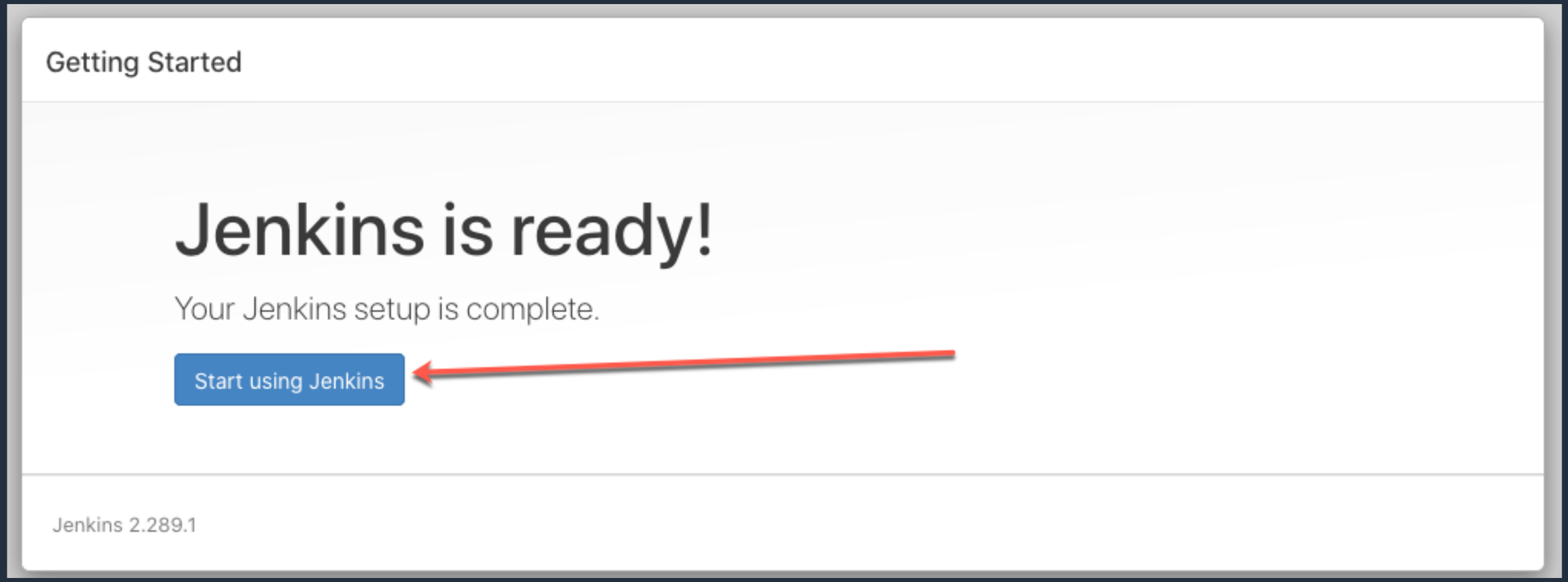
Save the Instance Configuration URL



- Jenkins URL must be resolvable (DNS can be used but must be resolvable)

aws

Congrats!



- Jenkins installation and initial configuration is complete!

aws

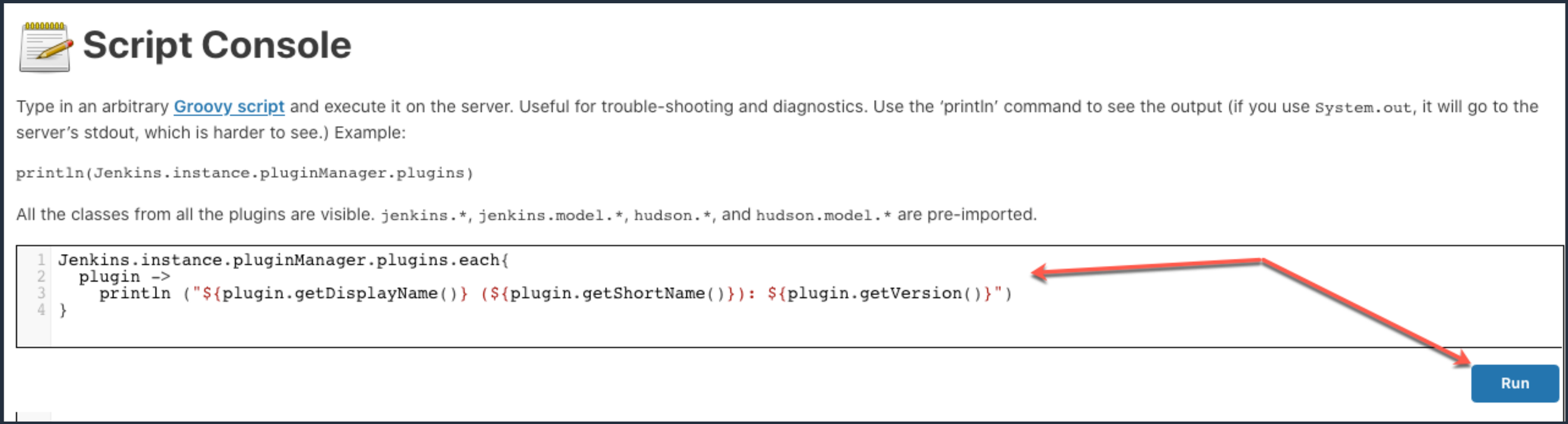
Jenkins Configuration

Basic Plugins and Configuration Settings

aws

Jenkins Plugins – how can I tell what plugins are installed?

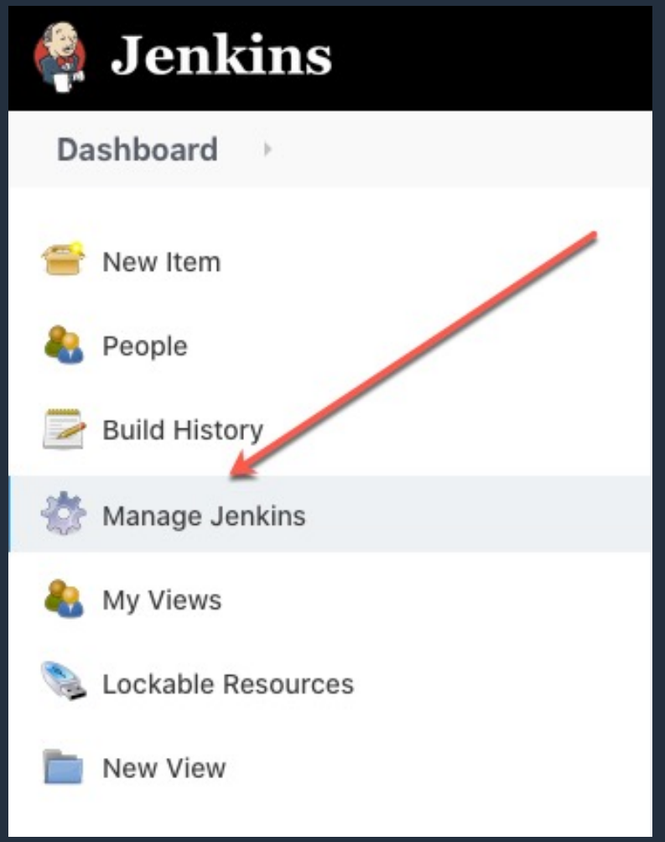
- Jenkins has over 1000+ plugins available!
- Plugins = Extension = You don't have to write code!!!
- Browse to http://{IP of Jenkins}:8080/script
- Run the following code In the Script Console:

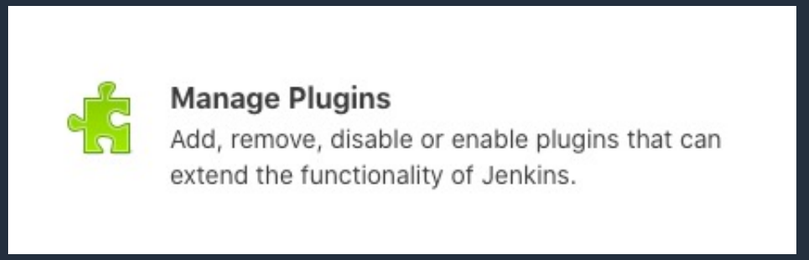


aws

Jenkins Plugins – Recommended for AWS and Connect

- Amazon Web Services SDK (aws-java-sdk)
- Authentication Tokens API Plugin
- Autofavorite for Blue Ocean
- BitBucket Branch Source Plugin
- Bitbucket Pipeline for Blue Ocean
- Blue Ocean Theme (blueocean)
- Blue Ocean Core JS
- Blue Ocean Pipeline Editor
- CloudBees AWS Credentials Plugin (aws-credentials)
- Common API for Blue Ocean
- Config API for Blue Ocean
- Convert to Pipeline (convert-to-pipeline)
- Dark Theme (dark-theme)
- Dashboard for Blue Ocean
- Design Language
- Display URL for Blue Ocean
- Docker Commons Plugin
- Docker Pipeline
- Events API for Blue Ocean
- Favorite
- Git Pipeline for Blue Ocean
- Github Pipeline for Blue Ocean
- Groovy
- HTML Publisher Plugin
- HTTP Request Plugin
- Handy Uri Templates 2.x API Plugin
- JIRA Integration for Blue Ocean
- JWT for Blue Ocean
- Jira plugin
- Mercurial plugin
- Personalization for Blue Ocean
- Pipeline SCM API for Blue Ocean
- Pipeline implementation for Blue Ocean
- Pipeline: AWS Steps
- Pub-Sub “light” Bus
- REST API for Blue Ocean
- REST Implementation for Blue Ocean
- Server Sent Events (SSE) Gateway Plugin
- Slack Notifications Plugin
- Theme Manager
- Variant Plugin
- Web for Blue Ocean
- i18n for Blue Ocean

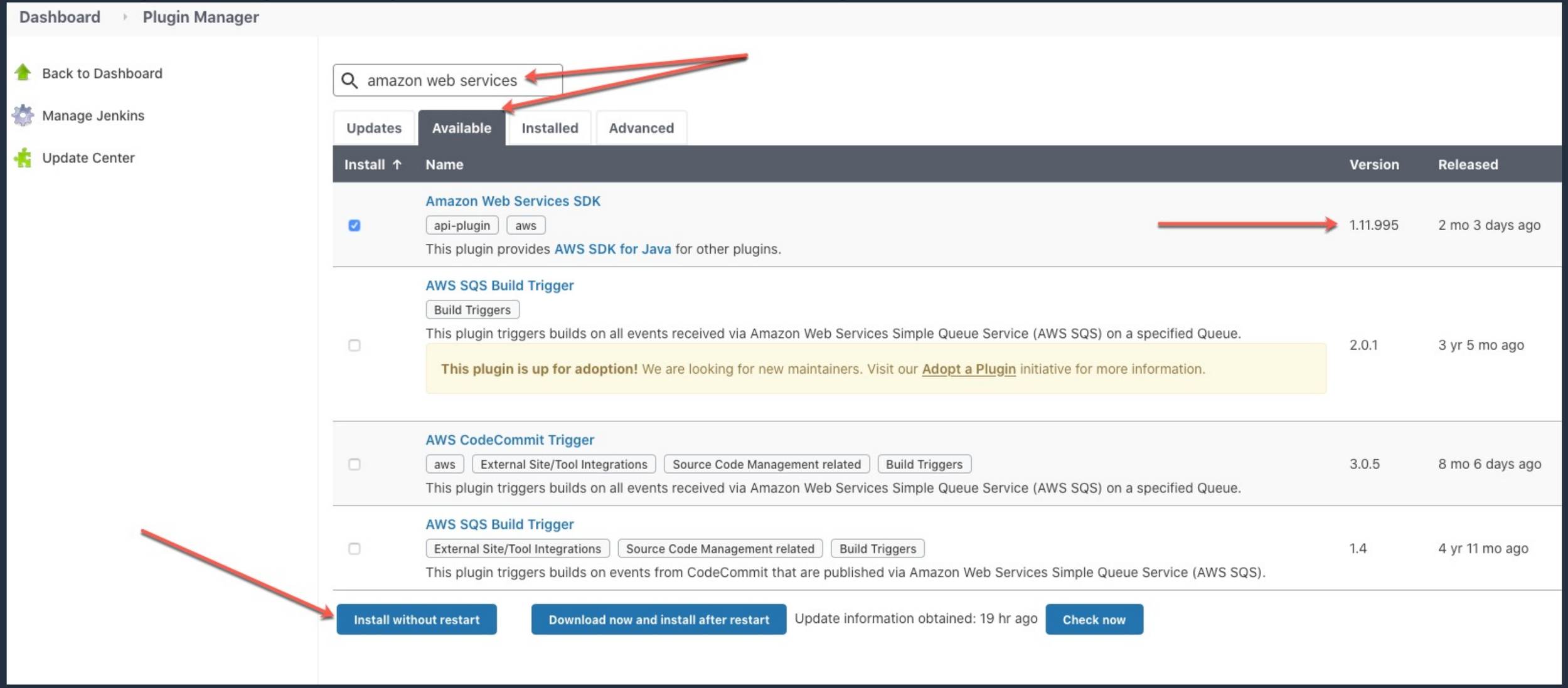




aws

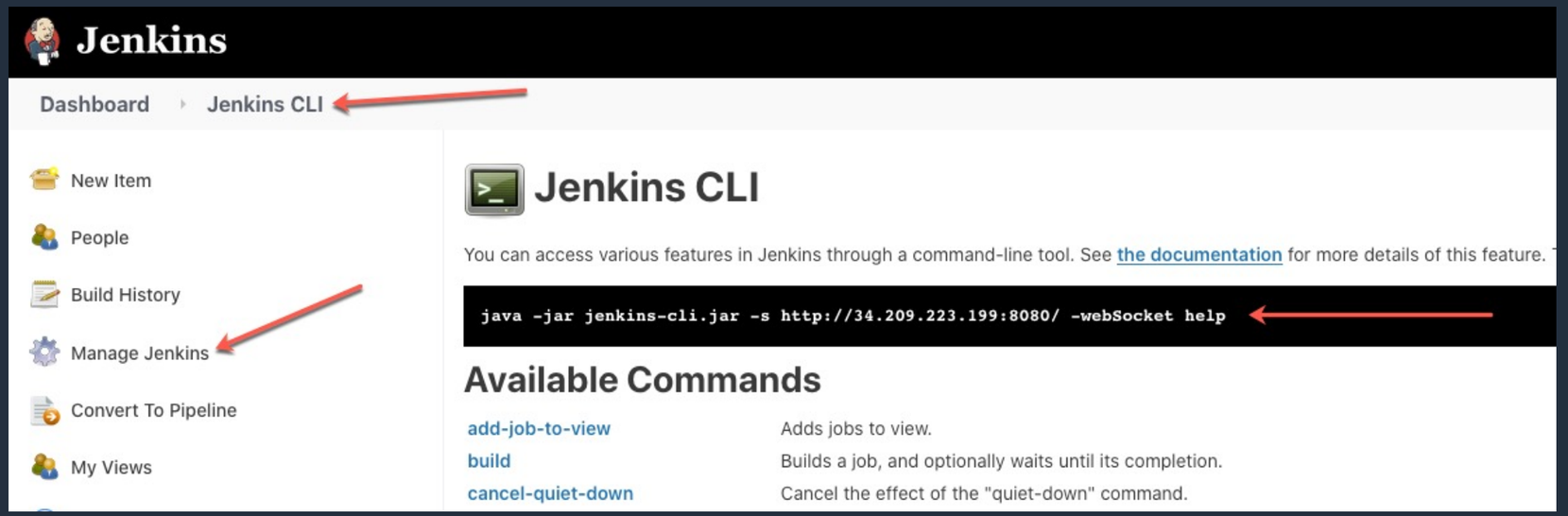
Jenkins Plugins – Installation

- Plugins Manager allows you to search, install, and update plugins



Jenkins Plugins – Installation via CLI Script

- Don’t spend 2 hours installing the plugins the hard way, use the CLI and a script!
- Download the JAR file (Manage Jenkins > Jenkins CLI) from the server for Command Line Access to Jenkins



- WGET the JAR file from the server

```
[ec2-user@ip-172-31-2-99 ~]$ wget http://localhost:8080/jnlpJars/jenkins-cli.jar
```

- Run the CLI commands to install ALL necessary plugins!

java -jar jenkins-cli.jar -auth {Username}:{Password} -s http://localhost:8080/ -webSocket install-plugin aws-java-sdk {plugin name(s) } -deploy -restart

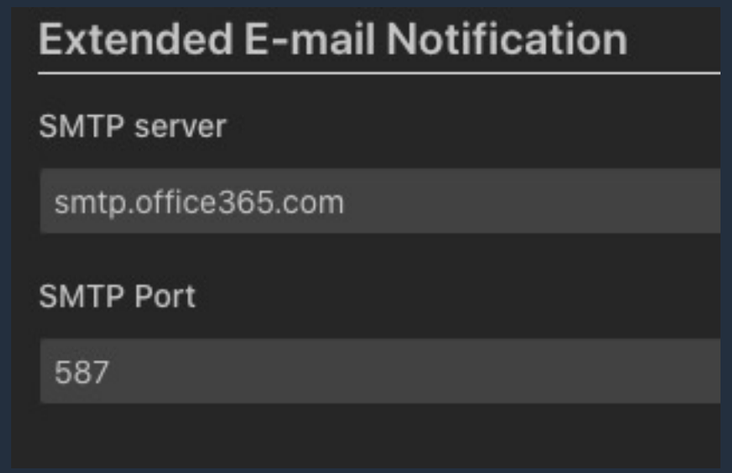
```
[ec2-user@ip-172-31-2-99 ~]$ java -jar jenkins-cli.jar -auth alhannah:            -s http://localhost:8080/ -webSocket install-plugin aws-java-sdk dark-theme authentication-tokens blueocean blueocean-core-js blueoc
ean-pipeline-editor blueocean-bitbucket-pipeline blueocean-autofavorite blueocean-dashboard blueocean-commons blueocean-config blueocean-display-url blueocean-events blueocean-git-pipeline blueocean-github-pipeline c
loudbees-bitbucket-branch-source aws-credentials convert-to-pipeline jenkins-design-language docker-commons docker-workflow favorite groovy http_request blueocean-jira jira mercurial pipeline-aws slack email-ext -dep
loy -restart
```

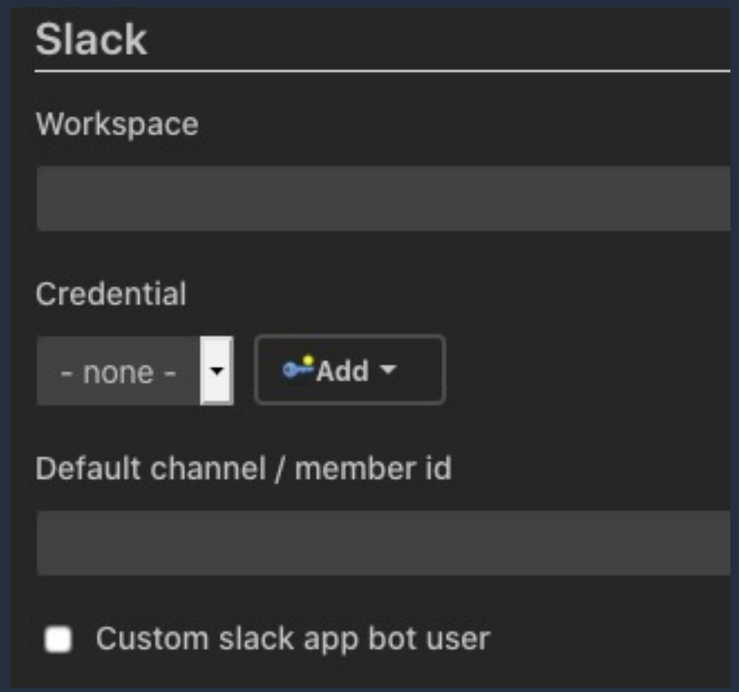

aws

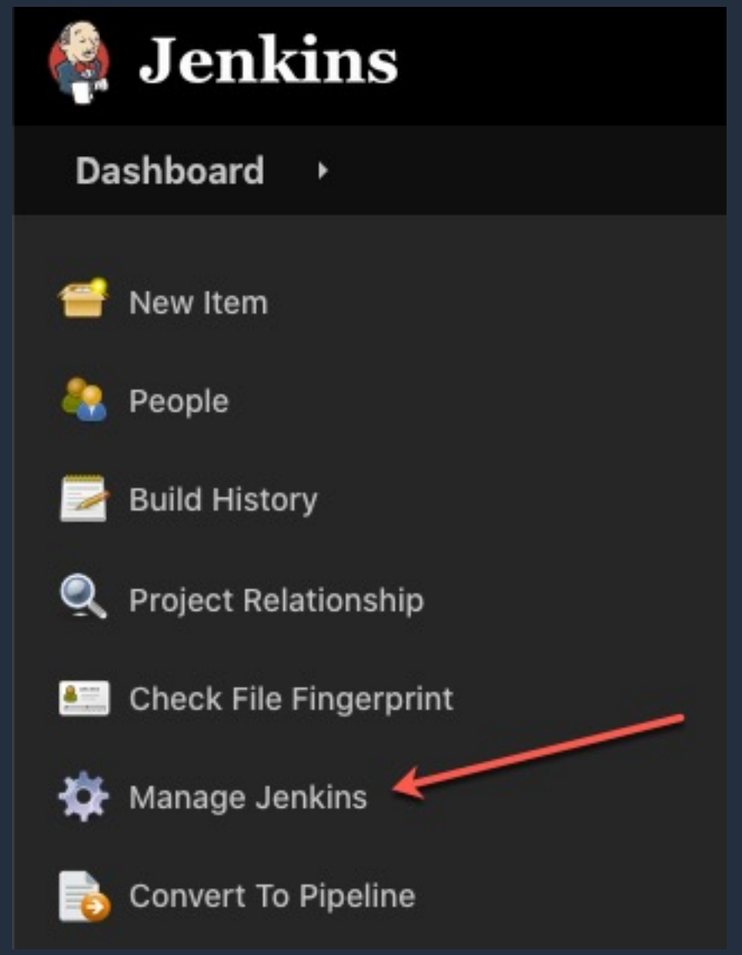
Jenkins Global Configuration

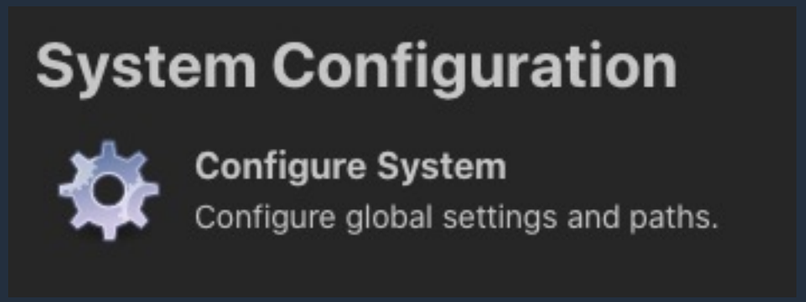
- High level server settings for Jenkins instance are required

- Jenkins URL
- System Admin email-address
- PATH Variables
- Theme
- Timestamper (Timestamps)
- Docker
- GitHub
- Groovy
- Extended E-mail Notification
- Slack





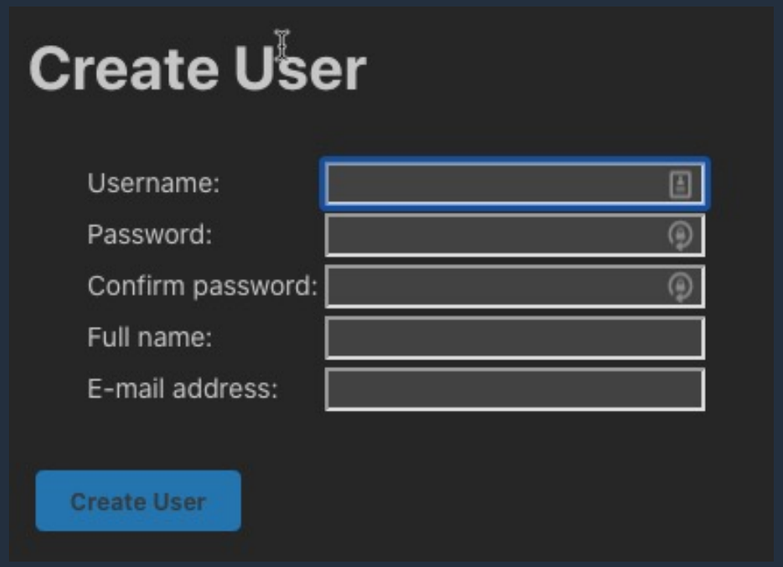


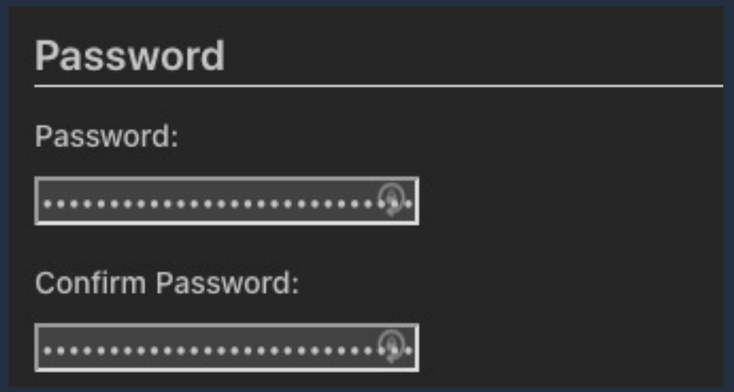


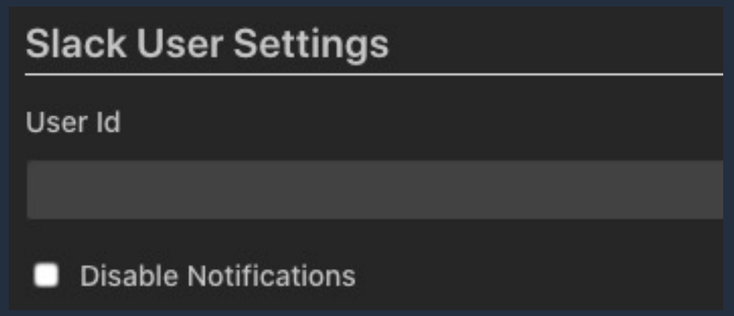
aws

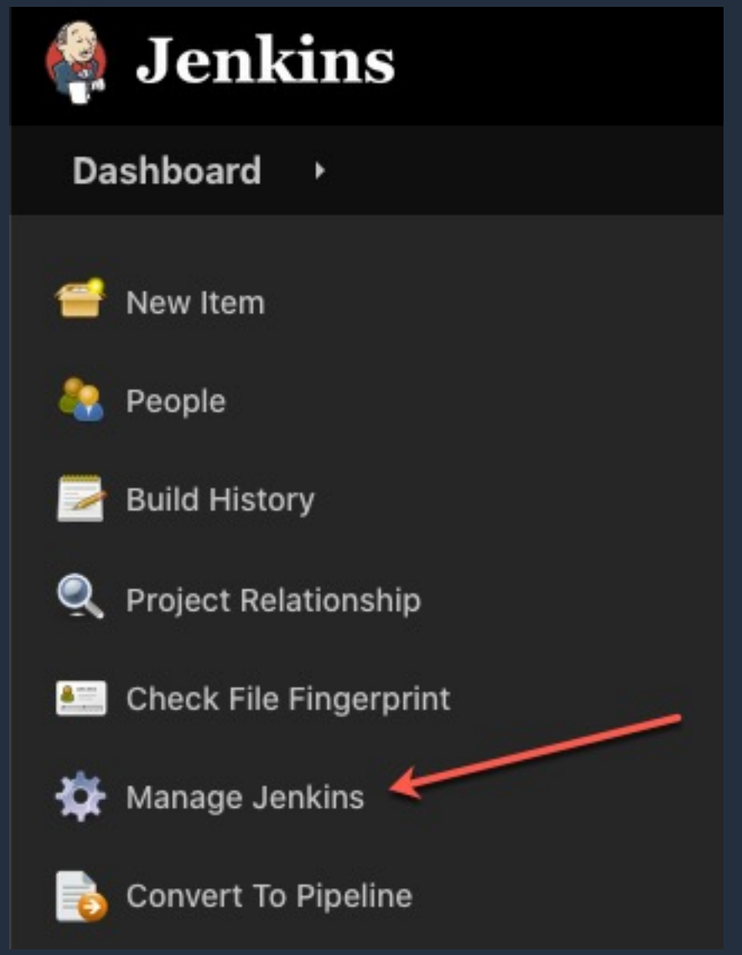
Jenkins Manage Users

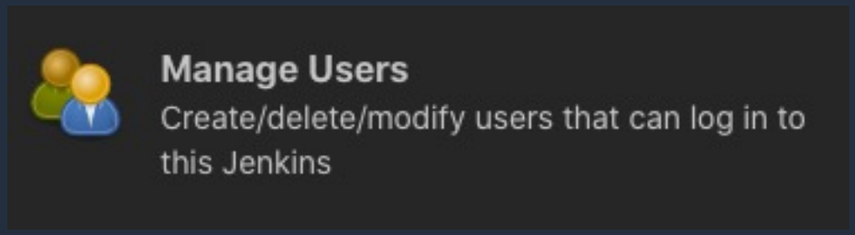
- Create and manage users for the Jenkins (GUI User Access):
- Username
- Full Name
- Email Address
- Password
- API Tokens (for REST API calls to Jenkins ONLY)
- Theme selection
- SSH Keys
- Slack ID
- Time zone







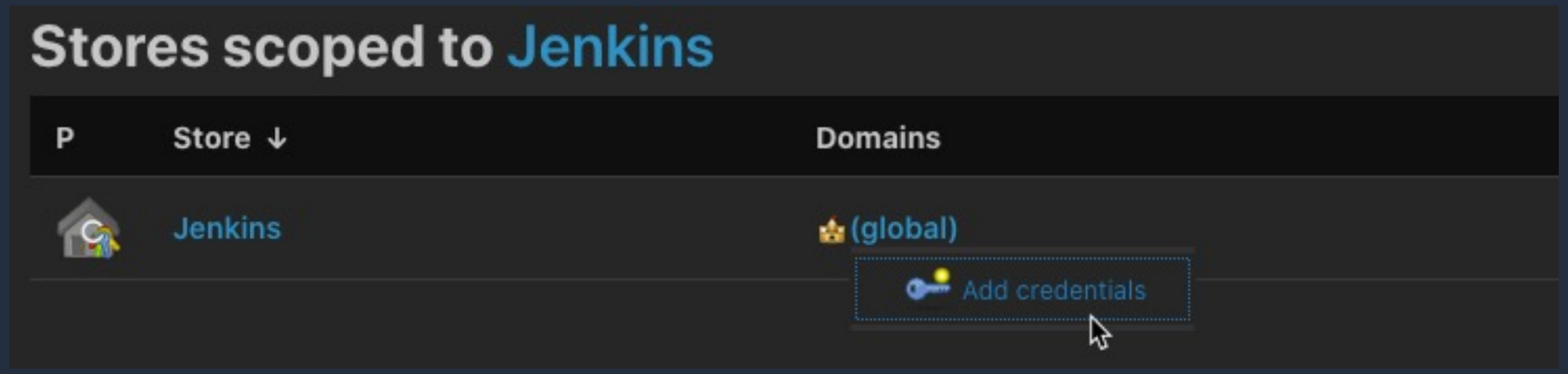


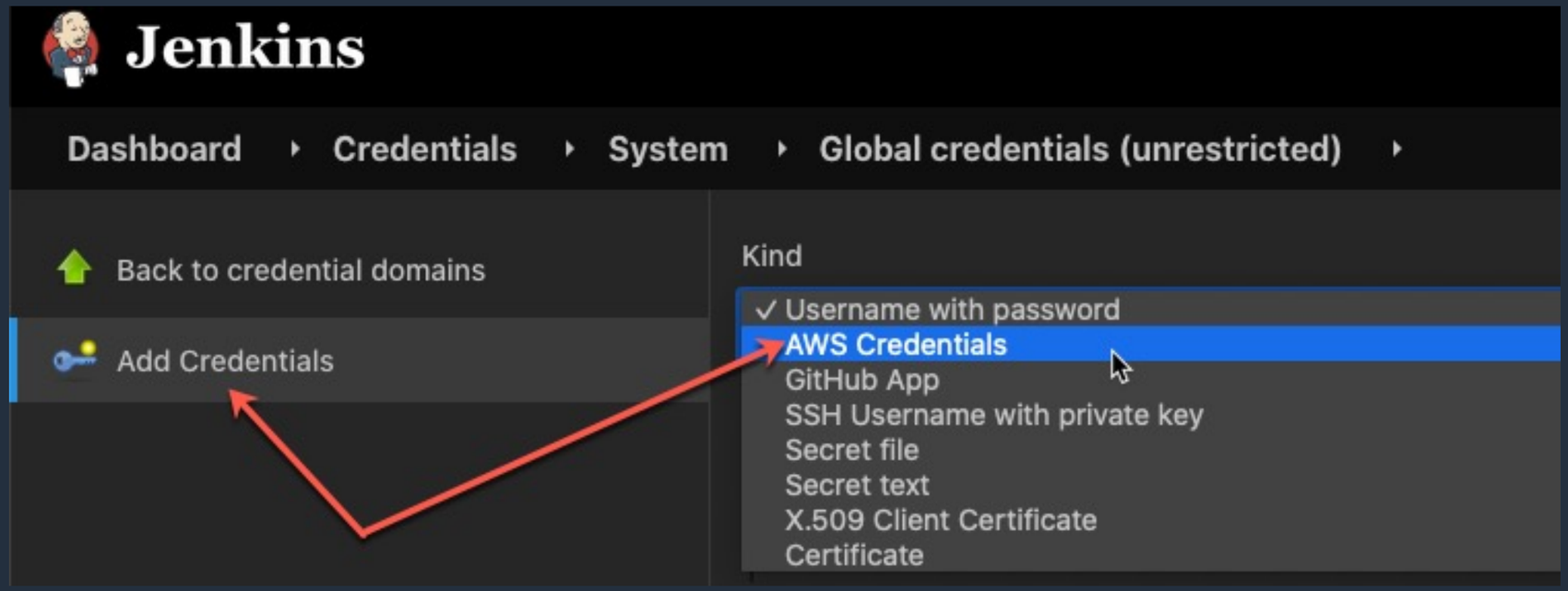


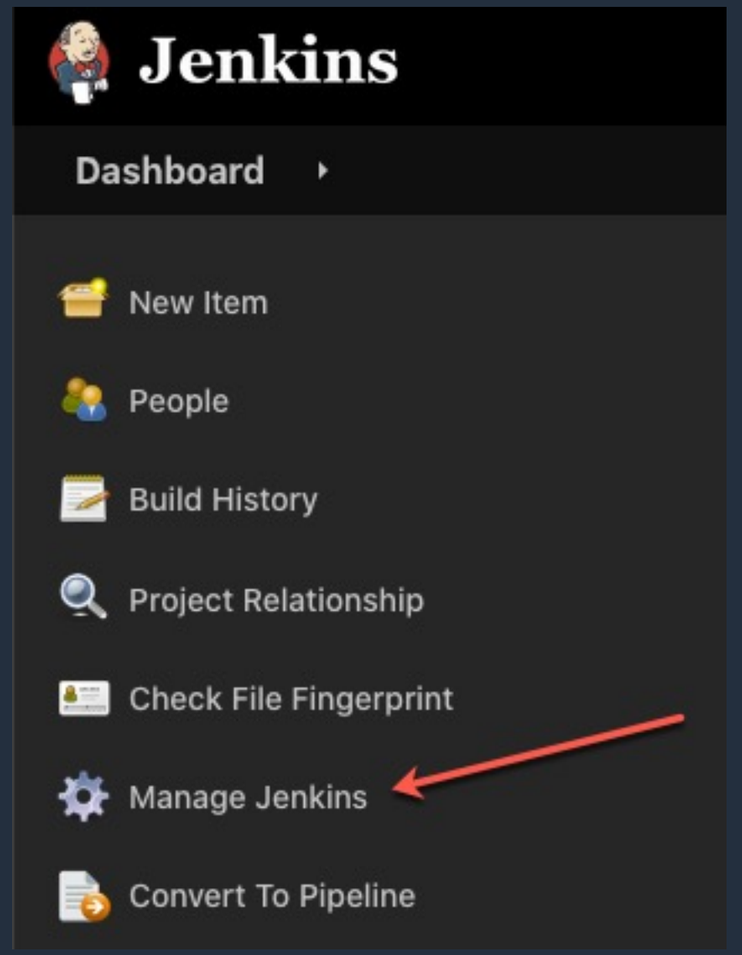
aws

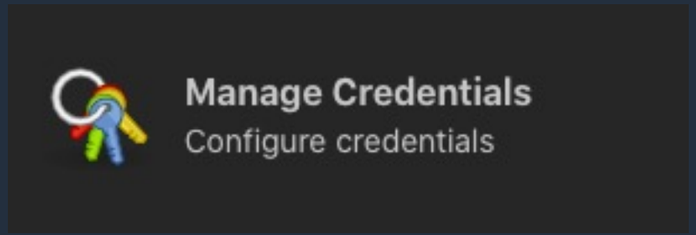
Jenkins Manage AWS API Keys

- Create and manage AWS API Keys in Jenkins (Pipeline Jobs):
- API Keys and Secret Keys to be abstract from developers and Pipeline Job Code



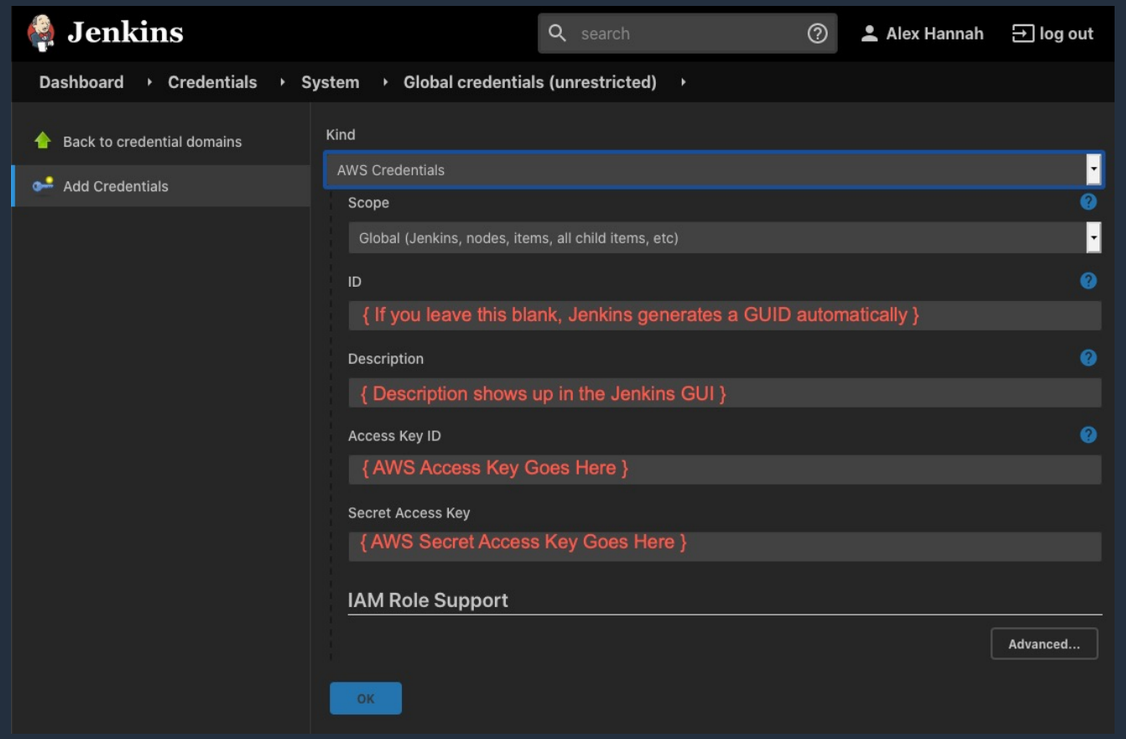






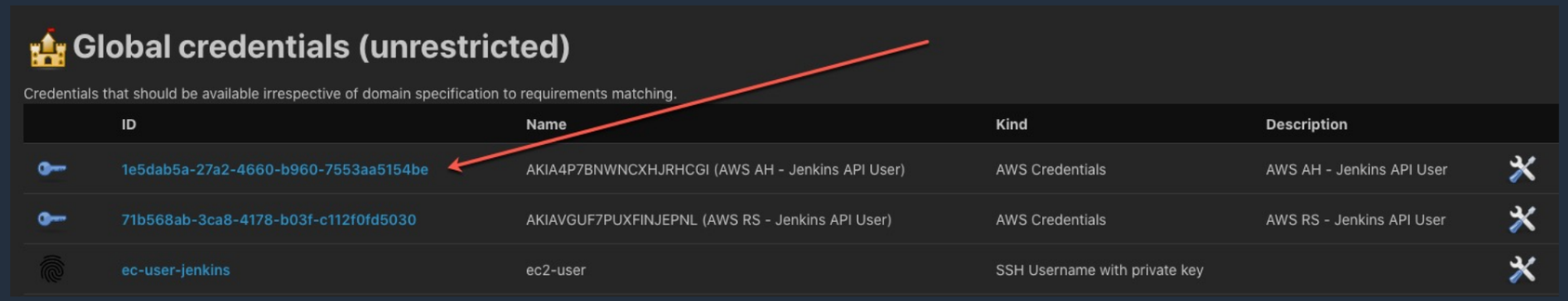
Jenkins Manage AWS API Keys (Continued)

- Create and manage AWS API Keys in Jenkins (Pipeline Jobs):
- API Keys and Secret Keys to be abstract from developers and Pipeline Job Code



Pipeline Code refers to Credentials on Jenkins using the “withAWS” syntax below!

```
stage('Create an Amazon Connect Instance'){
    steps {
        echo 'Creating the Amazon Connect Instance'
        withAWS(credentials: '71b568ab-3ca8-4178-b03f-c112f0fd5030', region: 'us-east-1') {
            // List all the Buckets
            script {
                //sh(script: "aws --version", returnStatus: true)
                def parsedJson =  sh(script: "aws connect create-instance --identity-management-type CONNECT_MANAGED
                echo "Instance details : ${parsedJson}"
                def instance = jsonParse(parsedJson)
                echo "ARN : ${instance.Arn}"
                ARN = instance.Arn
```



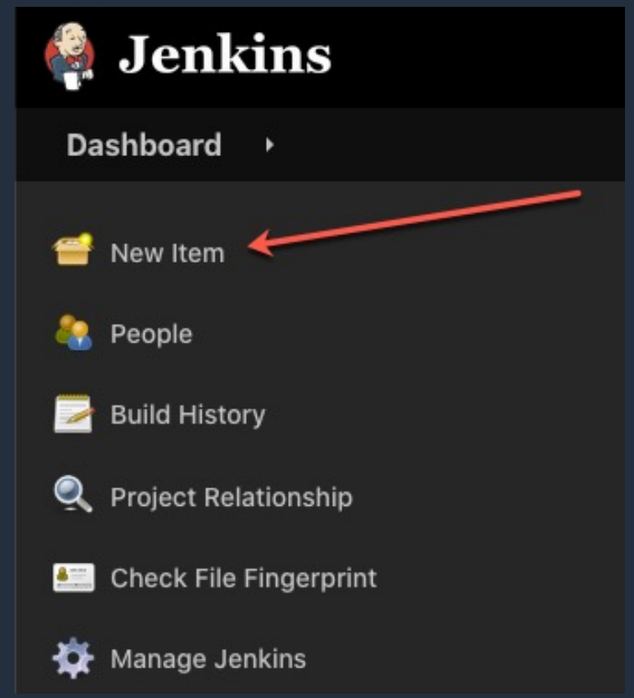
Jenkins Jobs

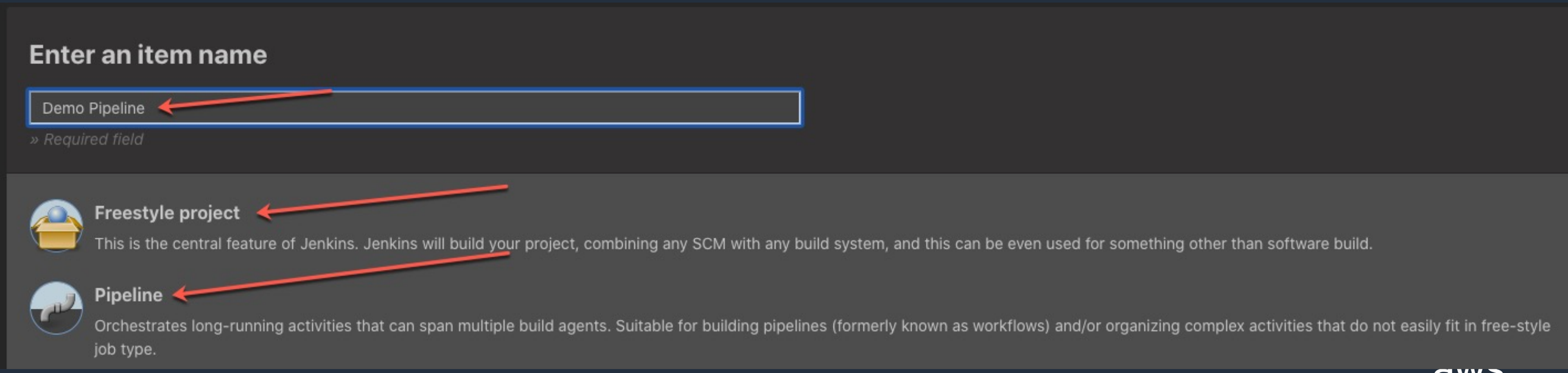
Pipeline Jobs for Amazon Connect

How do I create a Job in Jenkins?

- Jobs in Jenkins are found on the Dashboard under New Item
- Common Job types are Freestyle Projects or Pipelines
    - Pipelines tend to be more powerful than Freestyle Projects
- Pipelines can use two different languages:
    - Declarative Pipelines
    - Scripted Pipelines (Groovy Language)
    - Groovy Language is Java Based (similar to Node.js or Javascript)

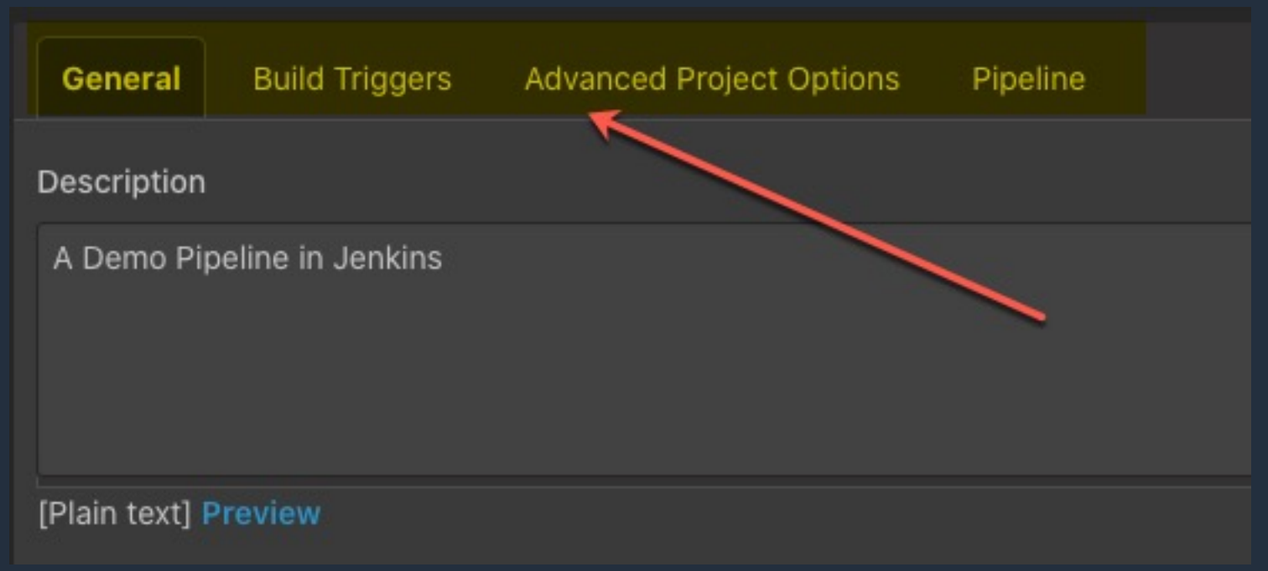
Note: We will focus on Declarative Pipelines here





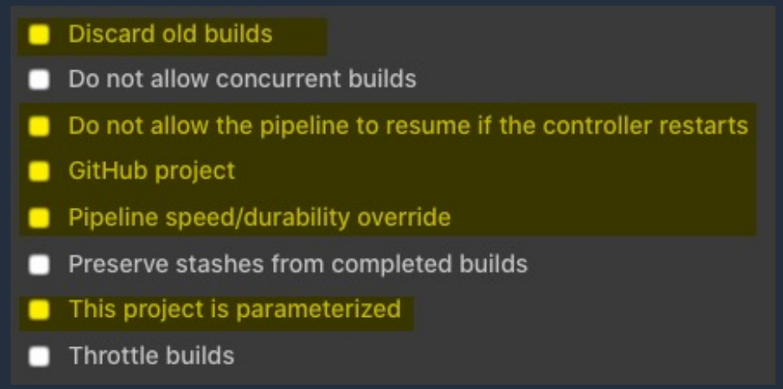
Jenkins Pipeline Job Options

- Single UI Panel with Tabs at the top
- Tabs represent major sections of the Pipeline:
    - General – General Job settings
    - Build Triggers – what triggers a job (SCM)
    - Advanced Project Options – Job name alias
    - Pipeline – Declarative Script to run

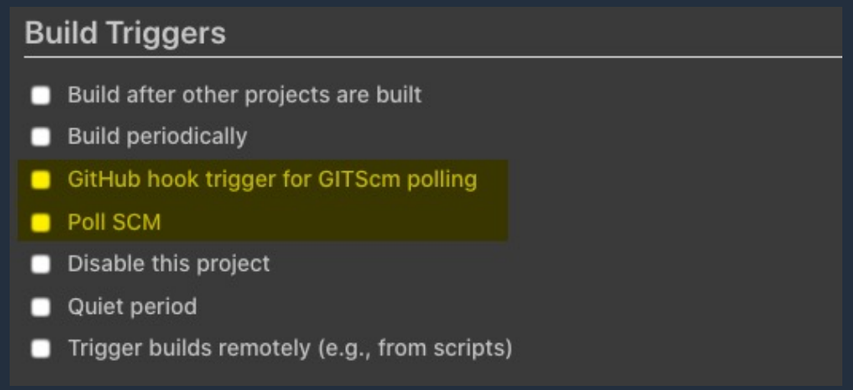


Save = Page Reload
Apply = Stay on Page

General Setting of Interest



Build Trigger Setting of Interest

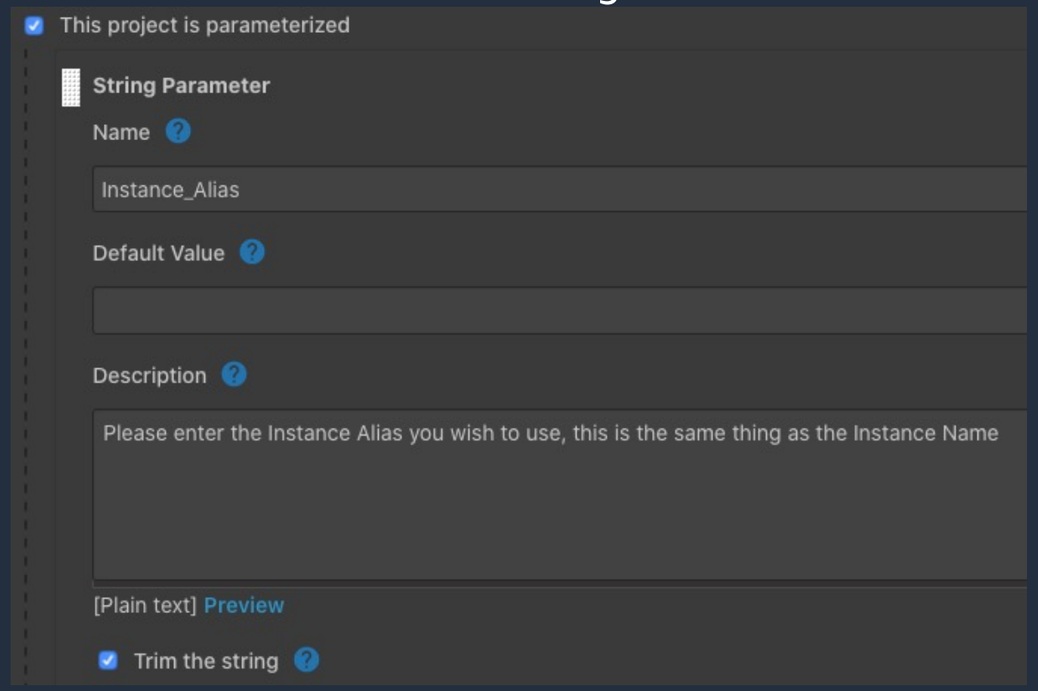


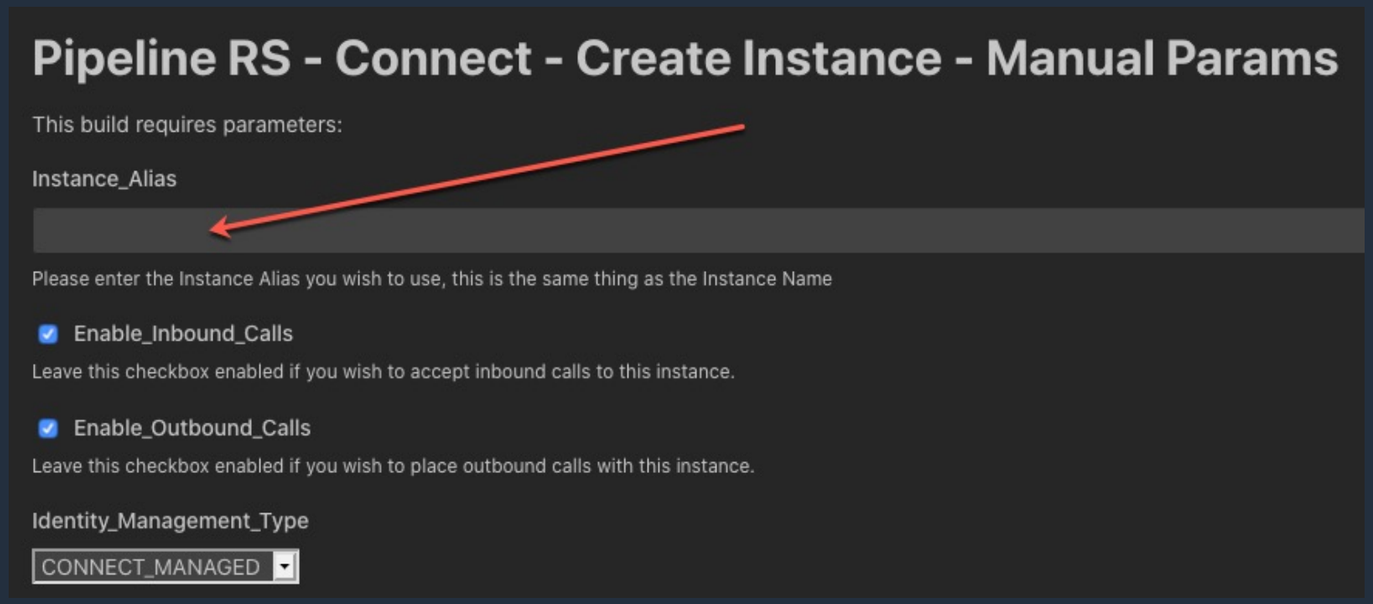
aws

Jenkins Pipeline Parameters

- Parameterized Projects allow you to supply options at runtime
- Parameters can be any of the following data types:
    - Strings
    - Booleans (Checkbox)
    - Choice (Pick list)
    - Credentials
    - Other types

Parameters at Design Time!

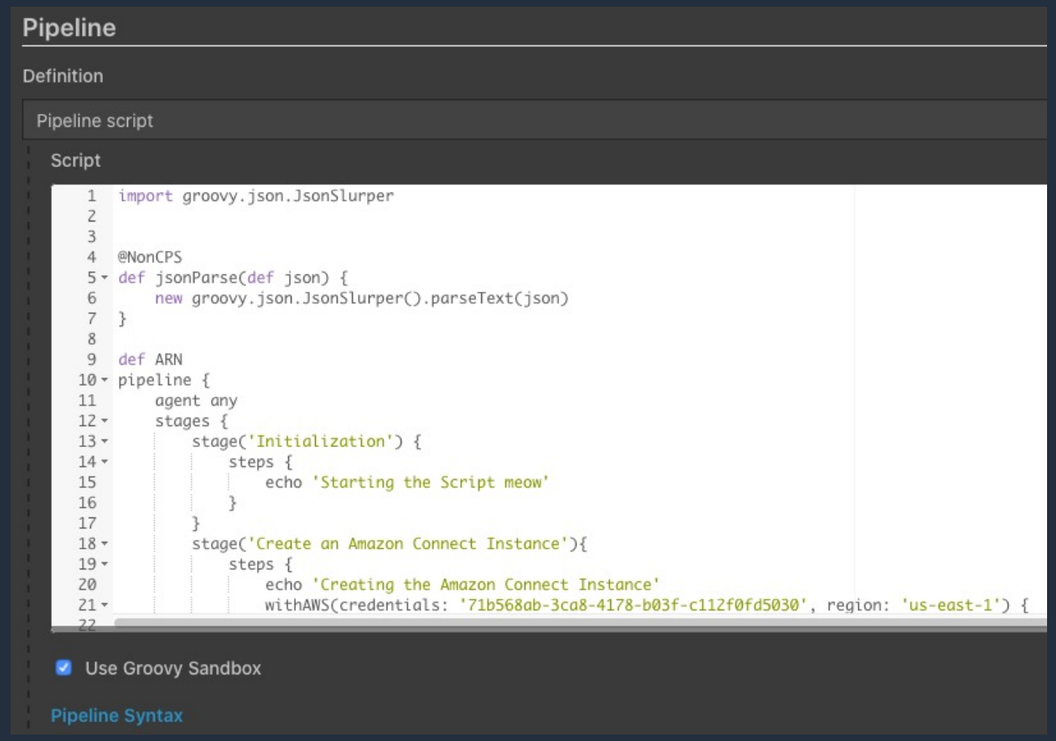




Parameters at Job Run Time!

Jenkins Pipeline Scripts

- Pipeline Scripts reside on the Jenkins Server
- Scripts are broken into Stages
- Each Stage can contain multiple Steps
- Global Environment variables can be used
- Post Actions (job cleanup actions) can be configured
    - Send an email notification
    - Create a message in a Slack Room



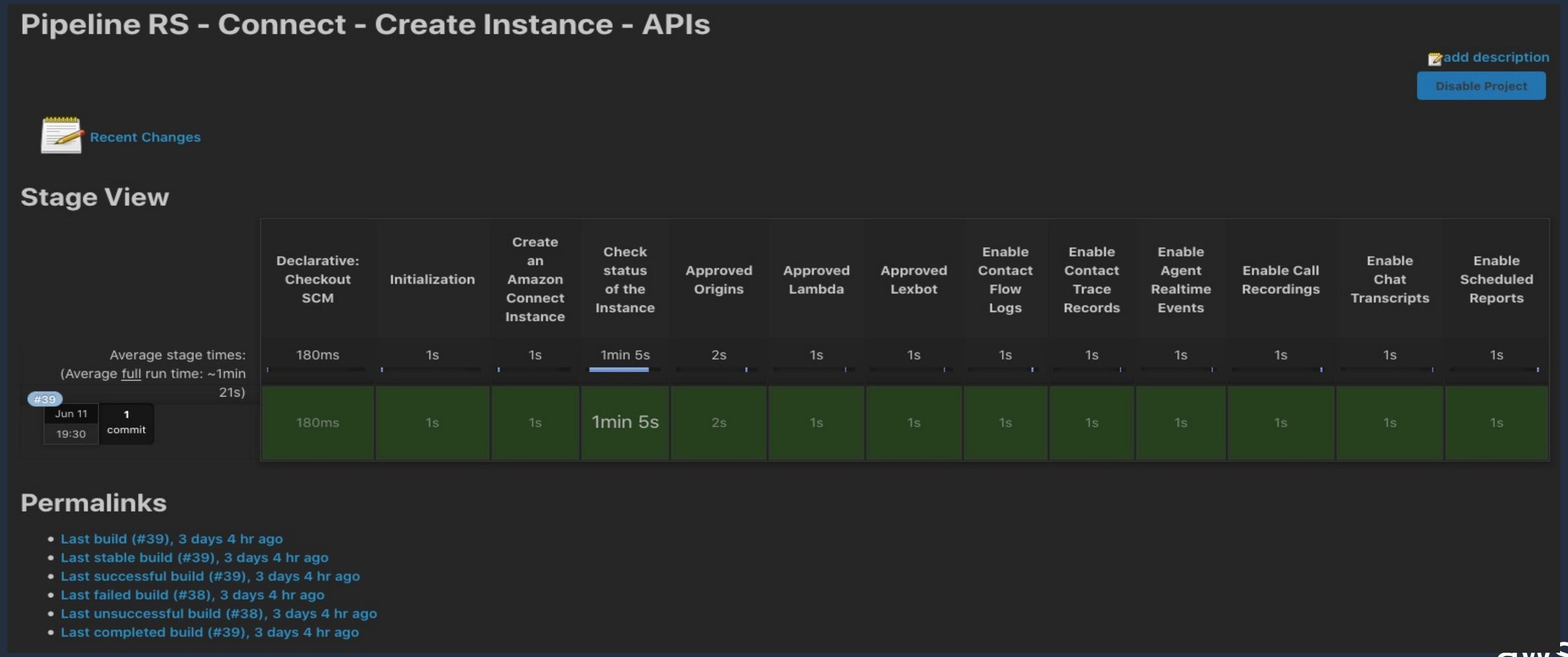

```
pipeline {
    agent any
    environment {
        //Contains Global variables and definitions
    }
    stages {
        stage('Step 1 in the Job'){
            //Perhaps pull down Github repo
        }
        stage('Step 2 in the Job'){
            //Perhaps create a Connect Instance
        }
        stage('Step 3 in the Job'){
            //Update Connect Instance settings
        }
    }
    post {
        //What to do after the job completes
        //Example: Email or Slack notification
    }
}
```

Structure of a Jenkins File

aws

Jenkins Job Output (Stages and Logs)

- Jenkins Jobs produce a Stage View
- Jobs also create stage logs

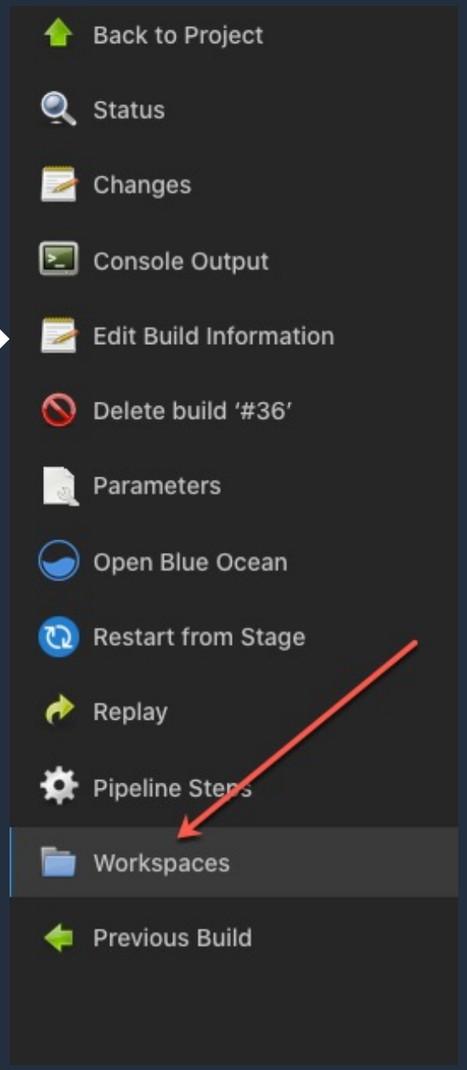


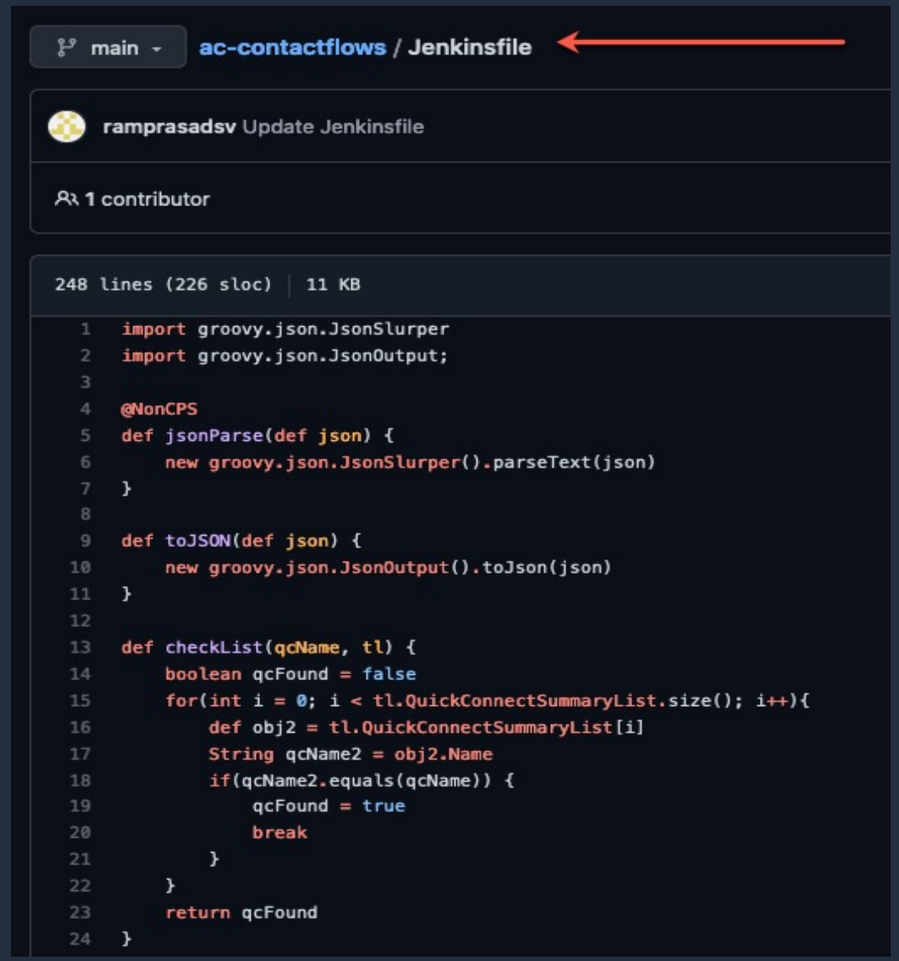
JenkinsFile and Workspace Location Non-GitHub vs GitHub Jobs

- Jenkins files, scripts, and “builds” run on the server in a “Workspace” folder during run time for Non-GitHub Jobs
- With GitHub integrations, the JenkinsFile is pulled from a GitHub Repo

```
pipeline {
    agent any
    environment {
        //Contains Global variables and definitions
    }
    stages {
        stage('Step 1 in the Job'){
            //Perhaps pull down Github repo
        }
        stage('Step 2 in the Job'){
            //Perhaps create a Connect Instance
        }
        stage('Step 3 in the Job'){
            //Update Connect Instance settings
        }
    }
    post {
        //What to do after the job completes
        //Example: Email or Slack notification
    }
}
```

Structure of a Jenkins File





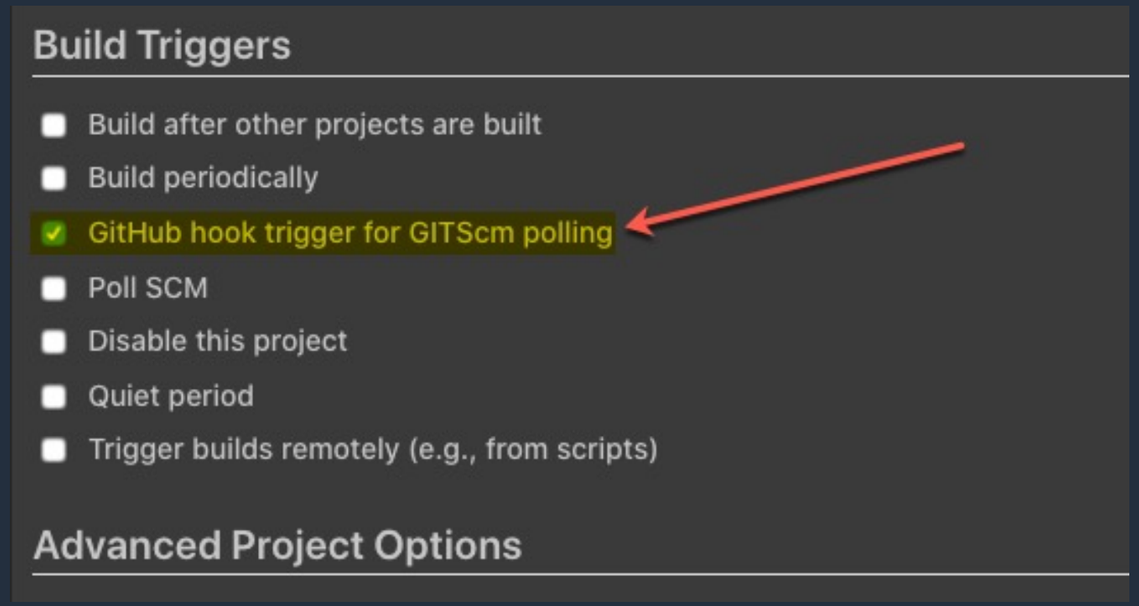

```groovy
import groovy.json.JsonSlurper
import groovy.json.JsonOutput;

@NonCPS
def jsonParse(def json) {
    new groovy.json.JsonSlurper().parseText(json)
}

def toJSON(def json) {
    new groovy.json.JsonOutput().toJson(json)
}

def checkList(qcName, tl) {
    boolean qcFound = false
    for(int i = 0; i < tl.QuickConnectSummaryList.size(); i++){
        def obj2 = tl.QuickConnectSummaryList[i]
        String qcName2 = obj2.Name
        if(qcName2.equals(qcName)) {
            qcFound = true
            break
        }
    }
    return qcFound
}
```

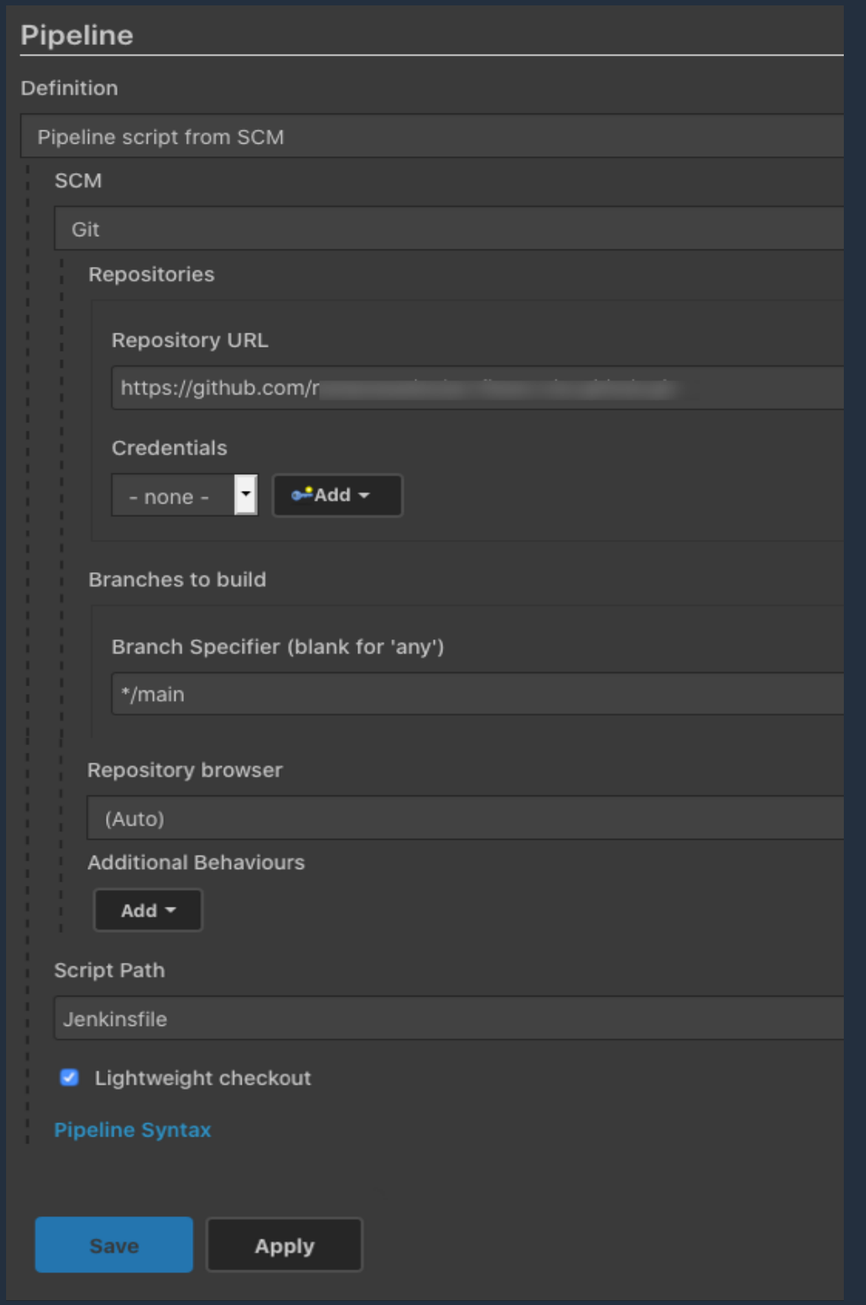
Default Path on Server:
/var/lib/jenkins/workspace

aws

Jenkins Pipelines + Github

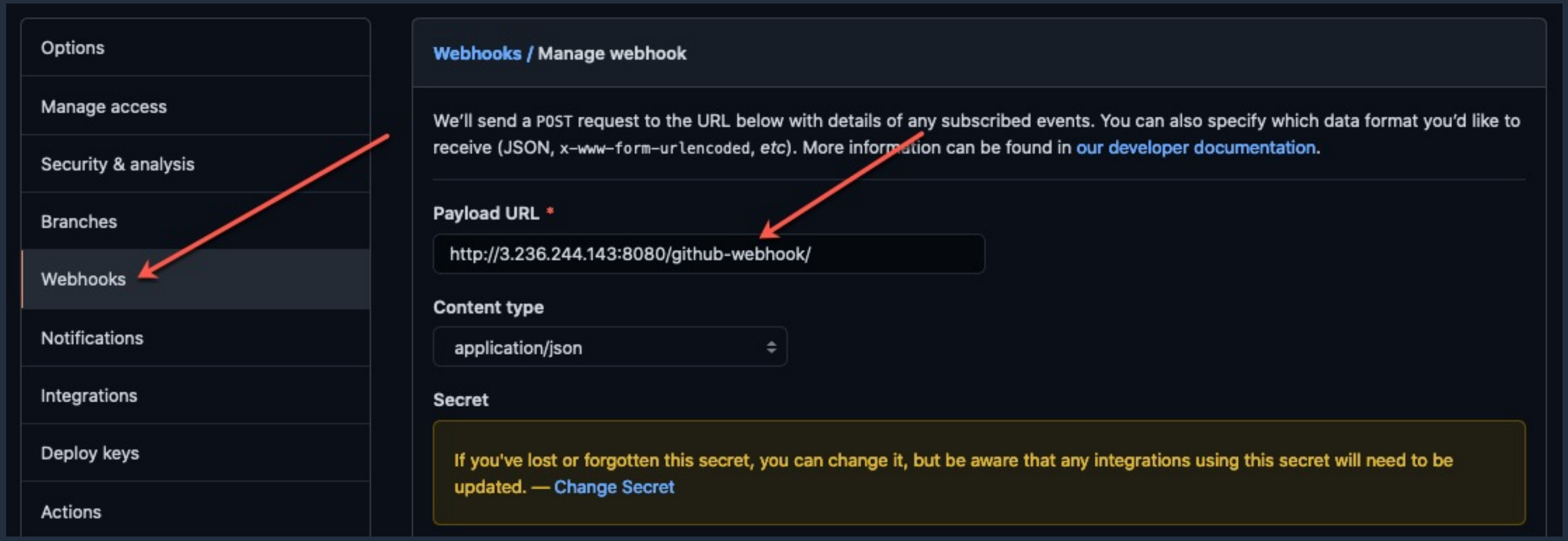
- Pipelines can either Poll or be Pushed to by GitHub repo
- Demo will showcase a GitHub Webhook notifying Jenkins on a Git Commit
- Pipelines can utilize different branches
- Github configuration can use authentication to non-public repo's





Jenkins Pipelines + Github Webhooks

- Pipelines can be executed by a webhook push by GitHub on a Commit to a repo
- Configured under GitHub > Settings > Webhooks

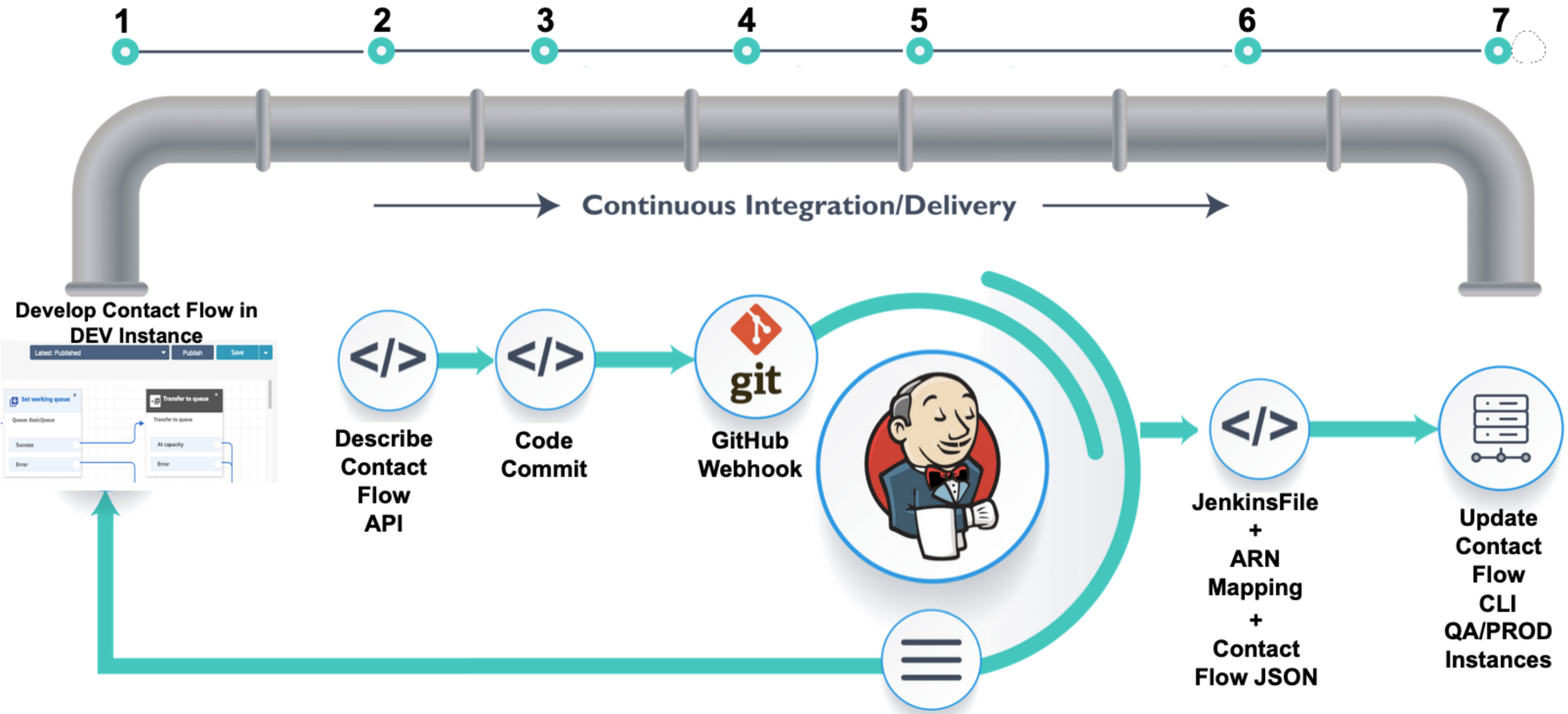


aws

Amazon Connect – CI/CD

Understanding the CI/CD Workflow

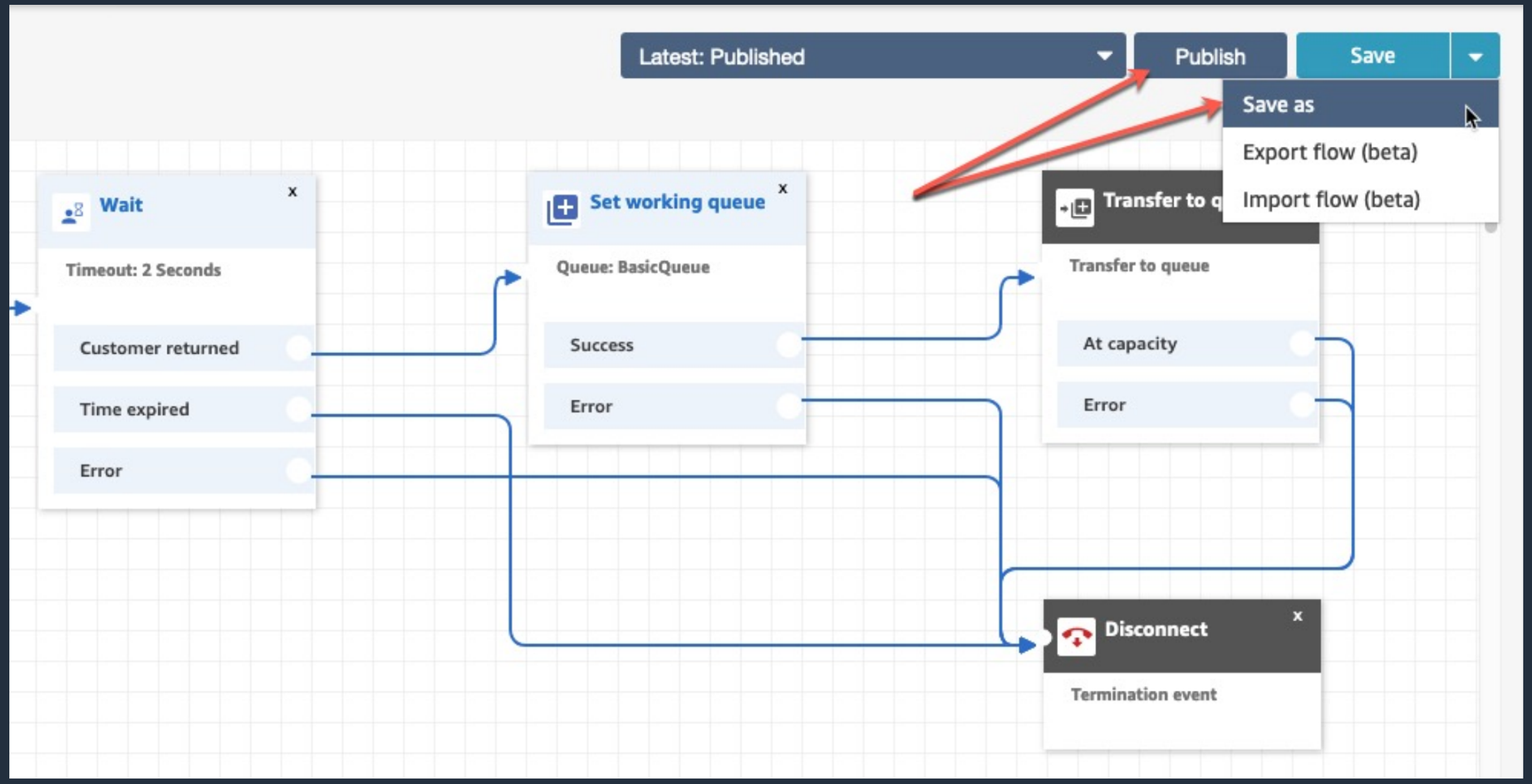
Amazon Connect CI/CD Workflow



aws

Develop and Save Contact Flows

- First step in the CI/CD Pipeline is to edit/configure your contact flows in your DEV environment
- Save/Save As and Publish to commit your changes in Amazon Connect



Question…. Why aren’t we using “Export flow (beta)?

DescribeContactFlow API

POSTMAN

- DescribeContactFlow API is called to output JSON of Contact Flow (Contact Flow Scripting Language)
- Postman can be configured to and utilized (saves time too!)

Amazon Connect / Contact Flows / **Describe Contact Flow**

Save

GET https://{{service}}.{{region}}.{{awsurl}}/contact-flows/{{instanceID}}/{{contactFlowID}} Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (11) Test Results Status: 200 OK Time: 205 ms Size: 3.62 KB Save Response

Pretty Raw Preview Visualize JSON

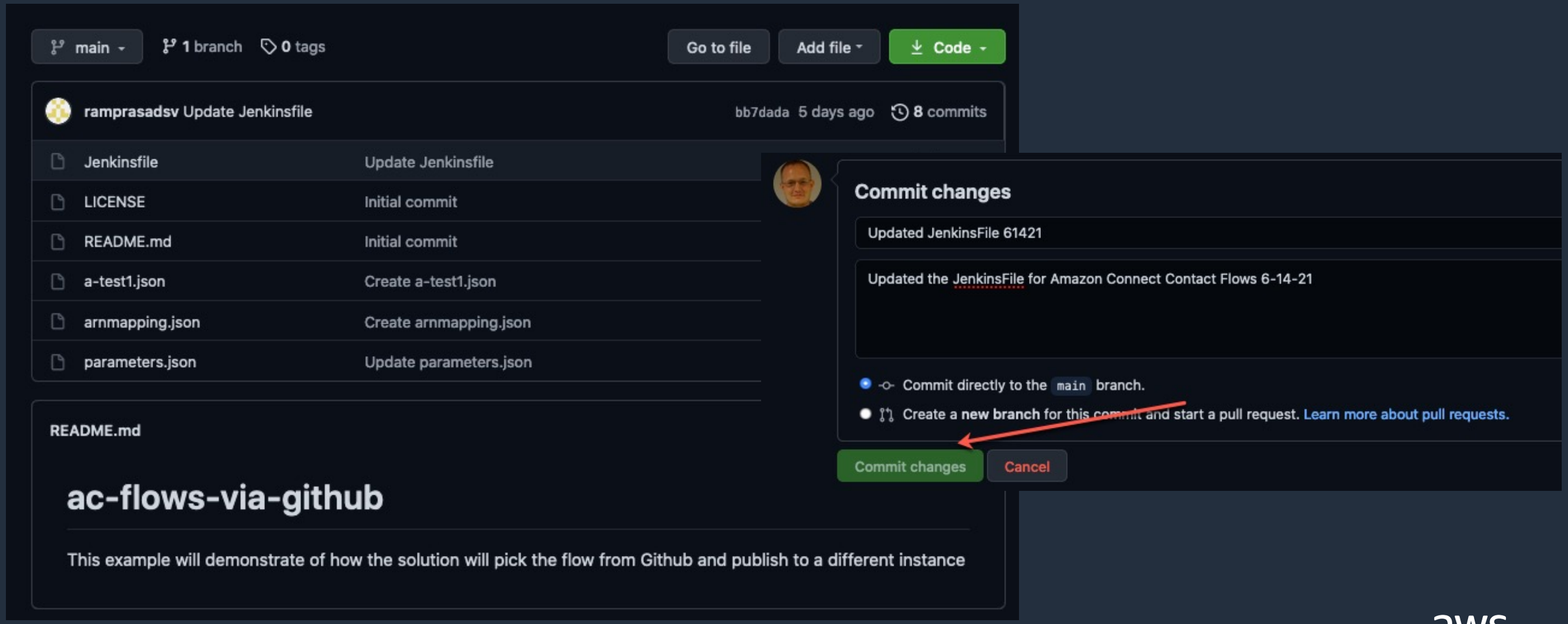

```
{
    "ContactFlow": {
        "Arn": "arn:aws:connect:us-east-1:858929345349:instance/8e56a433-8801-49b0-a825-144f24a682e3/contact-flow/7cd6f003-87e5-497e-ab67-b29e4f8241c7",
        "Id": "7cd6f003-87e5-497e-ab67-b29e4f8241c7",
        "Name": "_DEMO-Tracfone-BasicChatDisconnect-v1",
        "Type": "CONTACT_FLOW",
        "Status": null,
        "Description": "Tracfone Demo - Basic Chat Disconnect Flow v1",
        "Content": "{\"Version\":\"2019-10-30\",\"StartAction\":\"43f6ae81-9ea1-481b-b111-262fb53533bb\",\"Metadata\":{\"entryPointPosition\":{\"x\":20,\"y\":20},\"snapToGrid\":false,
            \"ActionMetadata\":{\"f5334f7a-e25e-4e12-8be2-937cbfc46d10\":{\"position\":{\"x\":822,\"y\":21},\"useDynamic\":false,\"queue\":
            {\"id\":\"arn:aws:connect:us-east-1:858929345349:instance/8e56a433-8801-49b0-a825-144f24a682e3/queue/457aff73-c6dd-45be-bd85-868b4a841824\",\"text\":\"BasicQueue\"}},
            \"8d761ecd-f0fd-4bfc-8afc-90fcddb9bf75\":{\"position\":{\"x\":1157,\"y\":20},\"useDynamic\":false},\"8e3b9777-83ef-4295-b82b-c0dc74a479c5\":{\"position\":{\"x\":1158,\"y\":300}},
            \"6eed77cf-d3ff-4419-853f-ddacbf9b9b9c\":{\"position\":{\"x\":502,\"y\":23},\"useDynamic\":false,\"timeoutUnit\":{\"display\":\"Seconds\",\"value\":\"second\"}},
            \"43f6ae81-9ea1-481b-b111-262fb53533bb\":{\"position\":{\"x\":220,\"y\":21},\"useDynamic\":false}}},\"Actions\":[{\"Identifier\":\"f5334f7a-e25e-4e12-8be2-937cbfc46d10\",
            \"Parameters\":{\"QueueId\":\"arn:aws:connect:us-east-1:858929345349:instance/8e56a433-8801-49b0-a825-144f24a682e3/queue/457aff73-c6dd-45be-bd85-868b4a841824\"},\"Transitions\":
            {\"NextAction\":\"8d761ecd-f0fd-4bfc-8afc-90fcddb9bf75\",\"Errors\":[{\"NextAction\":\"8e3b9777-83ef-4295-b82b-c0dc74a479c5\",\"ErrorType\":\"NoMatchingError\"}],\"Conditions\":[]},
            \"Type\":\"UpdateContactTargetQueue\"},{\"Identifier\":\"8d761ecd-f0fd-4bfc-8afc-90fcddb9bf75\",\"Transitions\":{\"NextAction\":\"8e3b9777-83ef-4295-b82b-c0dc74a479c5\",\"Errors\":[
            {\"NextAction\":\"8e3b9777-83ef-4295-b82b-c0dc74a479c5\",\"ErrorType\":\"NoMatchingError\"},{\"NextAction\":\"8e3b9777-83ef-4295-b82b-c0dc74a479c5\",
            \"ErrorType\":\"QueueAtCapacity\"}],\"Conditions\":[]},\"Type\":\"TransferContactToQueue\"},{\"Identifier\":\"8e3b9777-83ef-4295-b82b-c0dc74a479c5\",
            \"Type\":\"DisconnectParticipant\",\"Parameters\":{},\"Transitions\":{}},{\"Identifier\":\"6eed77cf-d3ff-4419-853f-ddacbf9b9b9c\",\"Parameters\":{\"TimeLimitSeconds\":\"2\",
            \"Events\":[\"CustomerReturned\"]},\"Transitions\":{\"NextAction\":\"8e3b9777-83ef-4295-b82b-c0dc74a479c5\",\"Errors\":[{\"NextAction\":\"8e3b9777-83ef-4295-b82b-c0dc74a479c5\",
            \"ErrorType\":\"NoMatchingError\"}],\"Conditions\":[{\"NextAction\":\"8e3b9777-83ef-4295-b82b-c0dc74a479c5\",\"Condition\":{\"Operator\":\"Equals\",\"Operands\":[\"WaitCompleted\"]}}
            ,{\"NextAction\":\"f5334f7a-e25e-4e12-8be2-937cbfc46d10\",\"Condition\":{\"Operator\":\"Equals\",\"Operands\":[\"CustomerReturned\"]}}]},\"Type\":\"Wait\"},
            {\"Identifier\":\"43f6ae81-9ea1-481b-b111-262fb53533bb\",\"Parameters\":{\"Text\":\"The agent has disconnected\"},\"Transitions\":
            {\"NextAction\":\"6eed77cf-d3ff-4419-853f-ddacbf9b9b9c\",\"Errors\":[],\"Conditions\":[]},\"Type\":\"MessageParticipant\"}]}",
        "Tags": {}
    }
```


- https://docs.aws.amazon.com/connect/latest/APIReference/API_DescribeContactFlow.html

aws

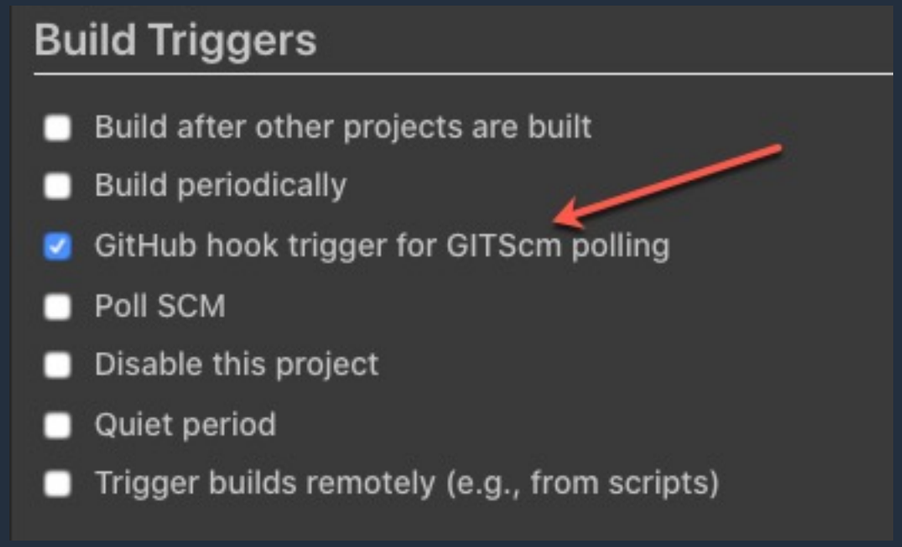
GitHub Commit: Save Contact Flows + JenkinsFile

- Upload your updated Amazon Connect Contact Flow to GitHub Repo
- Commit your JenkinsFile to GitHub Repo
- Remember: JenkinsFile executes the webhook in GitHub, which executes the Jenkins Job

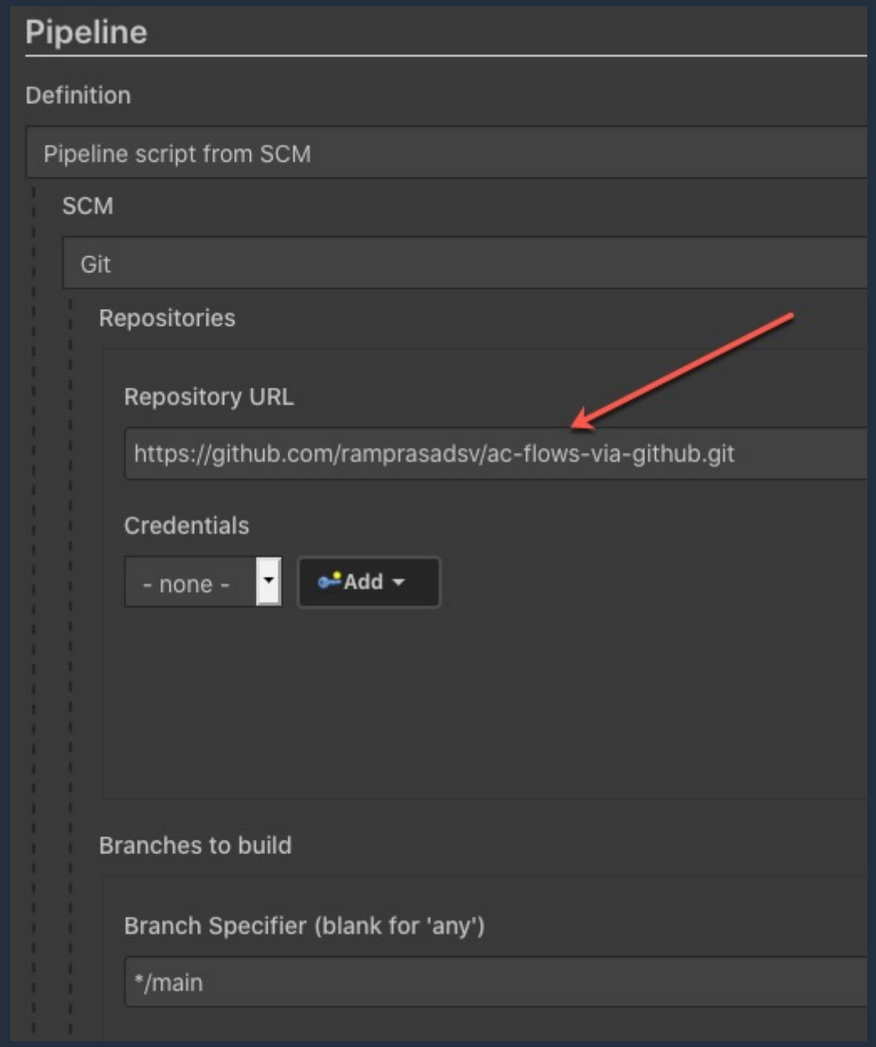


Jenkins Job + JenkinsFile

- Jenkins Server receives Webhook JSON payload from GitHub
- Payload instructs Jenkins there was a code commit and the repo name
- Jenkins uses the Job settings + GitHub Repo configuration to execute the appropriate Jenkins Job



+



aws

Amazon Connect – CI/CD

Demo 1: Instance Configuration

Amazon Connect – CI/CD

Demo 2: Contact Flow Synchronization

Amazon Connect – CI/CD

Demo 3: Quick Connect Synchronization

Amazon Connect – CI/CD

Demo 4: Queue Synchronization

Amazon Connect – CI/CD

Additional Resources

aws

Jenkins

Jenkins Homepage: https://www.jenkins.io/
Jenkins Download: https://www.jenkins.io/download/
Jenkins Docs: https://www.jenkins.io/doc/

Installing Jenkins on AWS EC2 Instance:
https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/

Installing Jenkins on AWS EC2 Video Walkthrough:
https://www.youtube.com/watch?v=jmm8DsosBqw

aws

Jenkins Training

Jenkins Training Courses: listed below are several training courses you may find of value, we utilized these during the development of the content you were shown today.

Getting Started with Jenkins:
https://app.pluralsight.com/library/courses/getting-started-jenkins/table-of-contents

Using Declarative Jenkins Pipelines:
https://app.pluralsight.com/library/courses/using-declarative-jenkins-pipelines/table-of-contents

Automating Jenkins with Groovy:
https://app.pluralsight.com/library/courses/automating-jenkins-groovy/table-of-contents

Jenkins Pipelines:
https://learn.acloud.guru/course/66271257-9c00-4bbf-8ec8-17da4a1b255f/dashboard

Jenkins Fundamentals:
https://learn.acloud.guru/course/e81aa3d0-bd3e-4a49-9bf8-dc4412e74f6c/dashboard

aws

GitHub Training

GitHub Training Courses: listed below are several training courses you may find of value, we utilized these during the development of the content you were shown today.

Getting Started with Github:

https://app.pluralsight.com/course-player?courseId=ff40abd2-2630-4c30-b1c0-d9f6498bb5fc

Source Control with Git:

https://learn.acloud.guru/course/104ff5d6-39c0-4116-b597-4d1bce0b8081/dashboard

Using SSH keys with Github:

https://www.youtube.com/watch?v=nQDFBd5NFA8

aws

GitHub Repo’s

Ram’s GitHub Repo:
https://github.com/ramprasadsv

Alex’s GitHub Repo:
https://github.com/Rodeorat316

aws

Q&A

Thank you!