



# 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

# Introduction to Jenkins

CI/CD for Amazon Connect



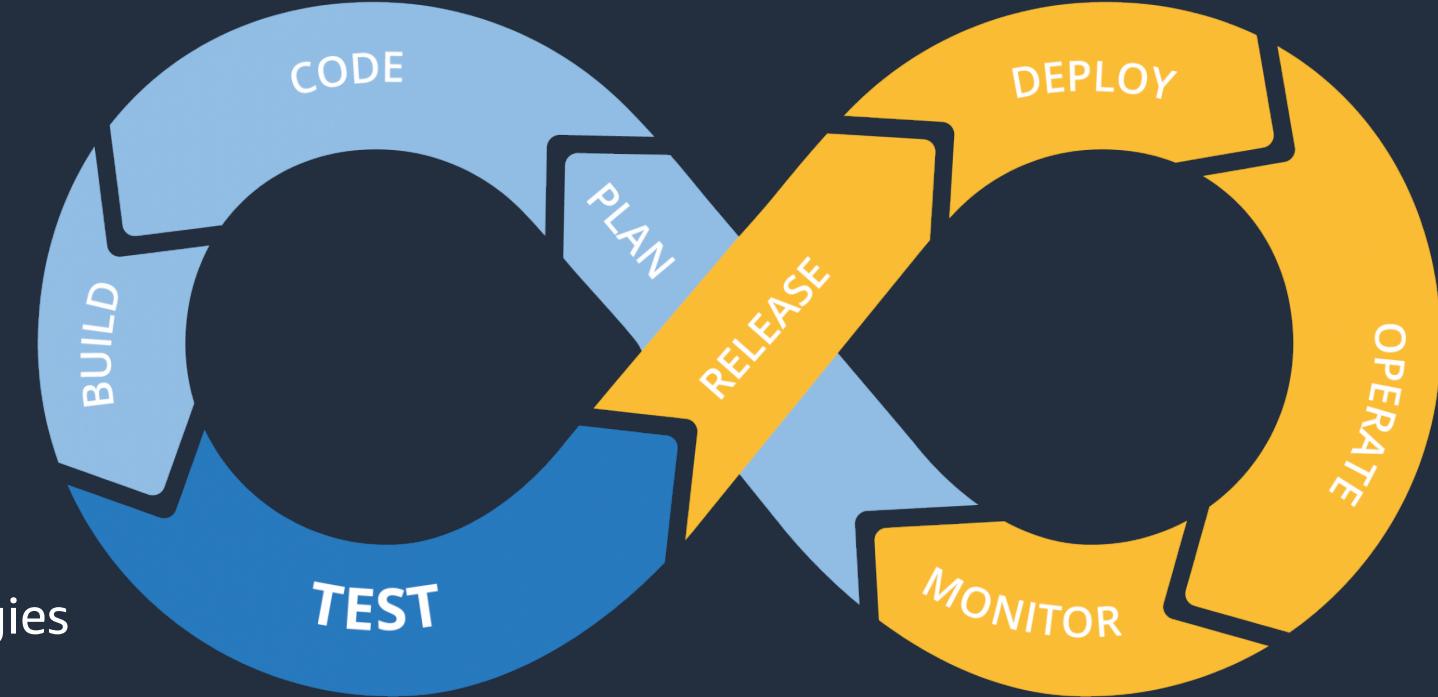
# So... what is Jenkins?

- 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?



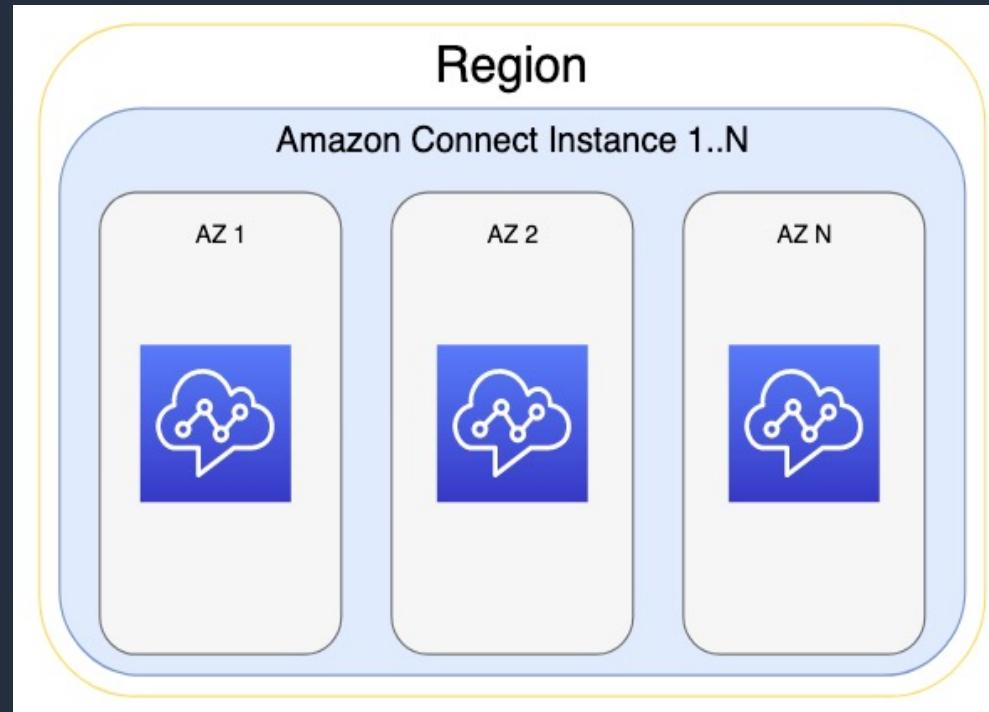
# What is CI/CD?

- Continuous Integration ( CI ) and Continuous Deliver ( CD )
- Principles 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 ).

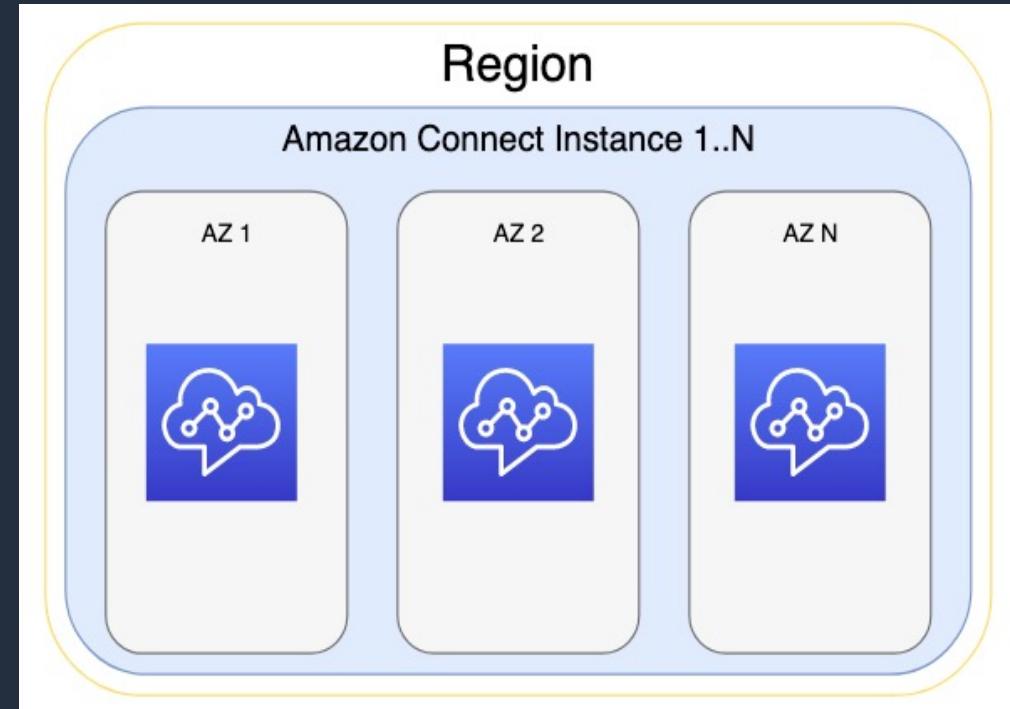


# Amazon Connect Instance Synchronization

Primary Instance – US-EAST-1



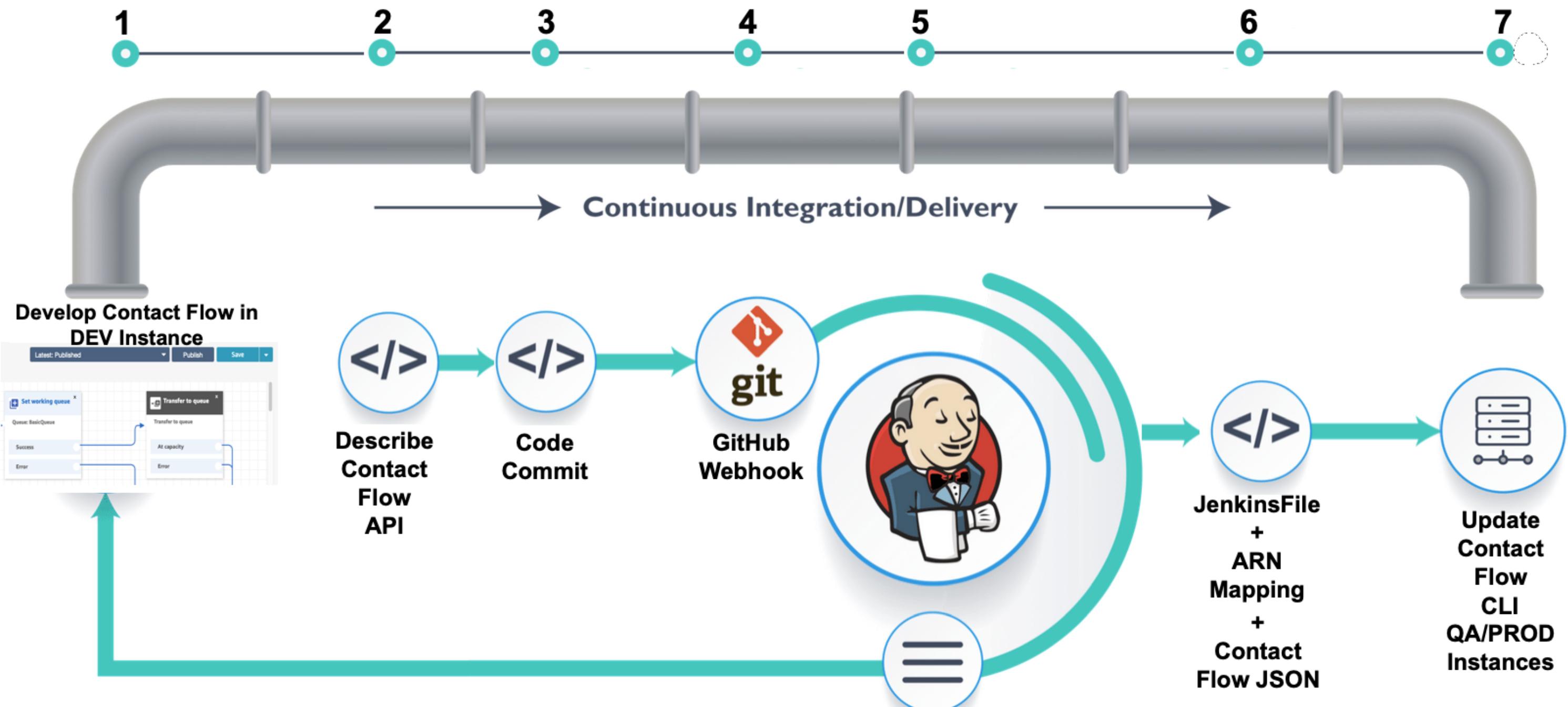
Secondary Instance – US-WEST-2



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?

# Amazon Connect CI/CD Workflow



# 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



# Step 1: Security Key Pairs

The screenshot shows the AWS Management Console homepage. The top navigation bar includes the AWS logo, a "Services" dropdown, a search bar with placeholder text "Search for services, features, marketplace products, and docs", and a keyboard shortcut "[Option+S]". Below the search bar, the title "AWS Management Console" is displayed. On the left, a sidebar titled "AWS services" lists "Recently visited services": Amazon Connect, CloudWatch, Lambda, Amazon Lex, IAM, S3, Billing, Amazon Honeycode, AWS Organizations, Amazon EventBridge, MediaConnect, DynamoDB, CloudFront, and Amazon Pinpoint. A red arrow points from the "EC2" icon in the "Recently visited services" list to the main content area. At the bottom of the sidebar, there is a "▶ All services" link.

The screenshot shows the "Network & Security" section of the AWS Management Console. It includes links for "Security Groups", "Elastic IPs", "Placement Groups", "Key Pairs" (which is highlighted with a yellow box and has a red arrow pointing to it), and "Network Interfaces". Below this, there is a "Load Balancing" section with links for "Load Balancers" and "Target Groups". A red arrow also points from the "Key Pairs" link in this menu to the corresponding item in the main content area.

- 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

The screenshot shows the AWS EC2 Key Pairs page. At the top right, there is a large orange button labeled "Create key pair". A thick red arrow points from the bottom left towards this button. The page has a search bar at the top left with placeholder text "Filter key pairs". Below the search bar, there are three columns: "Name", "Fingerprint", and "ID". The "Name" column has a downward arrow icon. The "Fingerprint" and "ID" columns have downward arrow icons. In the center, it says "No key pairs to display". At the top, there is a navigation bar with a "C" icon, "Actions ▾", and the "Create key pair" button.

- 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!

The screenshot shows the "Create key pair" wizard. At the top, it says "EC2 > Key pairs > Create key pair". The main section is titled "Key pair" with the sub-instruction: "A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance." Below this, there is a "Name" field containing "Jenkins Server SSH Keys" with the note: "The name can include up to 255 ASCII characters. It can't include leading or trailing spaces." There are two radio buttons for "File format": "pem" (selected) and "ppk". The "pem" option is described as "For use with OpenSSH". The "ppk" option is described as "For use with PuTTY". Below these, there is a "Tags (Optional)" section with three rows of tags:

- Key: KP, Value - optional: Jenkins EC2, Remove button
- Key: EC2 Server, Value - optional: Jenkins, Remove button
- Key: Region, Value - optional: US WEST, Remove button

A "Add tag" button is located below these rows. At the bottom, there are "Cancel" and "Create key pair" buttons.

# Saving the Key Pairs

✓ Successfully created key pair X

**Key pairs (1)**

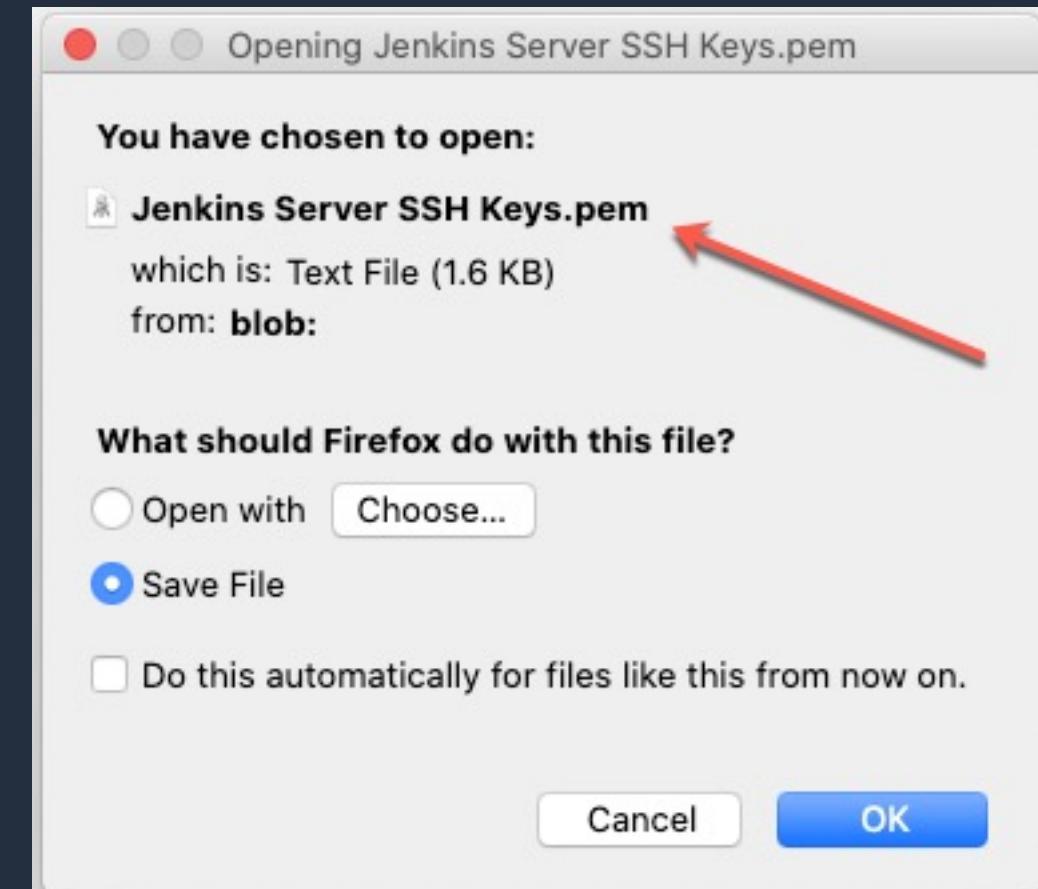
Actions Create key pair

Filter key pairs 1

<input type="checkbox"/>	Name	Fingerprint	ID
<input type="checkbox"/>	Jenkins Server SSH Keys	f1:95:e1:43:40:a4:82:9c:a9:70:70:98:c5...	key-0d72e7f65bd810db4

- 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 %
wllhann@38f9d3c82d1d Jenkins PEM File % chmod 400 JenkinsServer-SSHKeys.pem
wllhann@38f9d3c82d1d Jenkins PEM File %
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 %
```



# Step 2: Security Groups

AWS Management Console

AWS services

Recently visited services

- Amazon Connect
- CloudWatch
- Lambda
- Amazon Lex
- EC2
- IAM
- Billing
- AWS Organizations
- Amazon EventBridge
- S3
- Amazon Honeycode
- MediaConnect
- DynamoDB
- CloudFront
- Amazon Pinpoint

All services

Elastic Block Store

- Volumes
- Snapshots
- Lifecycle Manager

Network & Security

- Security Groups New
- Elastic IPs New
- Placement Groups
- Key Pairs
- Network Interfaces New

Load Balancing

- Load Balancers
- Target Groups New

- 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

# Creating the Security Group

Security Groups (1/1) <a href="#">Info</a>								<a href="#">Actions</a> <a href="#">▼</a>	<a href="#">Create security group</a>						
<input checked="" type="checkbox"/>	Name	▼	Security group ID	▼	Security group name	▼	VPC ID	▼	Description	▼	Owner	▼	Inbound rules count	▼	Outbound
<input checked="" type="checkbox"/>	-		sg-e26d53d1		default		vpc-c17f4cb9 <a href="#">Edit</a>		default VPC security gr...		858929345349		1 Permission entry		1 Permission

EC2 > Security Groups > Create security group

### Create security group [Info](#)

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

Security group name [Info](#)  
Jenkins Server Security Group  
Name cannot be edited after creation.

Description [Info](#)  
Jenkins Server Security Group

VPC [Info](#)  
vpc-c17f4cb9

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

Inbound rules <a href="#">Info</a>					
This security group has no inbound rules.					
<a href="#">Add rule</a>					
Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>	
SSH	TCP	22	My IP <a href="#">▼</a> 100.785.251/32 <a href="#">X</a>	Jenkins Inbound SSH	
HTTP	TCP	80	My IP <a href="#">▼</a> 100.785.251/32 <a href="#">X</a>	Jenkins Inbound HTTP Port 80	
Custom TCP	TCP	8080	My IP <a href="#">▼</a> 100.785.251/32 <a href="#">X</a>	Jenkins Inbound TCP Port 8080	

# 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

Jenkins Server Security Group

Security group ID

sg-095b8df85e4b793c7

Description

Jenkins Server Security Group

VPC ID

vpc-c17f4cb9 

Owner

858929345349

Inbound rules count

3 Permission entries

Outbound rules count

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!

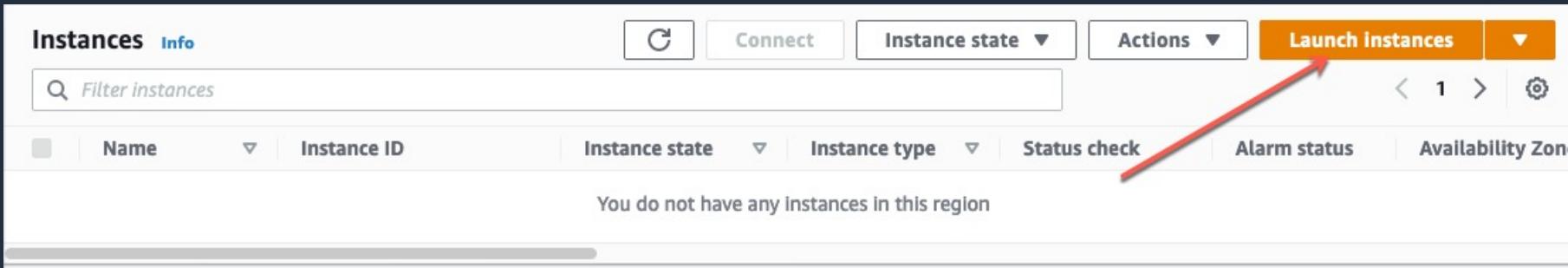
# Step 3: Launch the Instance

The screenshot shows the AWS Management Console homepage. The top navigation bar includes the AWS logo, a 'Services' dropdown, a search bar with placeholder text 'Search for services, features, marketplace products, and docs', and a keyboard shortcut '[Option+S]'. Below the header is the title 'AWS Management Console'. On the left, a sidebar titled 'AWS services' lists 'Recently visited services' such as Amazon Connect, CloudWatch, Lambda, and Amazon Lex, followed by a 'All services' link. In the main content area, several service icons are displayed in a grid, including EC2 (which is highlighted with a yellow box and has a red arrow pointing to it from the left), IAM, S3, Billing, AWS Organizations, Amazon EventBridge, Amazon Honeycode, MediaConnect, DynamoDB, CloudFront, and Amazon Pinpoint. At the bottom of the page is a footer with a copyright notice.

The screenshot shows the 'EC2 Dashboard' with a 'New' badge. The left sidebar has a tree view with 'Events', 'Tags', 'Limits', and a expanded 'Instances' node. Under 'Instances', the 'Instances' link is highlighted with an orange box and has a red arrow pointing to it from the top right. Other options in this section include 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances' (with a 'New' badge), 'Dedicated Hosts', 'Scheduled Instances', and 'Capacity Reservations'. The right side of the dashboard contains a large, light-gray area with a faint 'Get Started' message.

- 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)

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

1. Choose AMI   2. Choose Instance Type   3. Configure Instance   4. Add Storage   5. Add Tags   6. Configure Security Group   7. Review

Cancel and Exit

Search for an AMI by entering a search term e.g. "Windows"

Search by Systems Manager parameter

Quick Start

My AMIs

Amazon Linux Free tier eligible

Community AMIs

Free tier only i

**Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0800fc0fa715fdcfe (64-bit x86) / ami-04071714ae6bc5aa7 (64-bit Arm)**

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.

Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

**macOS Big Sur 11.4 - ami-01eacf319ba97c72**

The macOS Big Sur AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Root device type: ebs   Virtualization type: hvm   ENA Enabled: Yes

Select 64-bit (x86) 64-bit (Arm)

Select 64-bit (Mac)

A screenshot of the 'Step 1: Choose an Amazon Machine Image (AMI)' wizard. The 'Amazon Linux' entry is highlighted with a yellow background and a red arrow pointing to the 'Select' button. The 'Select' button for the 'Amazon Linux' entry is highlighted with a blue border. The 'Select' button for the 'macOS Big Sur' entry is also visible at the bottom right.

# 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!!!

1. Choose AMI    2. Choose Instance Type    3. Configure Instance    4. Add Storage    5. Add Tags    6. Configure Security Group    7. Review

### Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families ▾    Current generation ▾    Show/Hide Columns

Currently selected: t2.medium (- ECUs, 2 vCPUs, 2.3 GHz, -, 4 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t3	t3.large	2	8	EBS only	Yes	Up to 5 Gigabit	Yes

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Instance Details](#)

# Assigning Instance Details – Public IP

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

1. Choose AMI    2. Choose Instance Type    **3. Configure Instance**    4. Add Storage    5. Add Tags    6. Configure Security Group    7. Review

### Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

**Number of instances**  Launch into Auto Scaling Group [i](#)

**Purchasing option**  Request Spot instances

**Network**  [C](#) Create new VPC

**Subnet**  [C](#) Create new subnet

**Auto-assign Public IP**  Use subnet setting (Enable)  Enable  Disable

**Placement group**  Add instance to placement group

**Capacity Reservation**

**Domain join directory**  [C](#) Create new directory

**IAM role**  [C](#) Create new IAM role

**Shutdown behavior**

**Stop - Hibernate behavior**  Enable hibernation as an additional stop behavior

**Enable termination protection**  Protect against accidental termination

**Monitoring**  Enable CloudWatch detailed monitoring  
Additional charges apply.

**Tenancy**   
Additional charges will apply for dedicated tenancy.

**Elastic Inference**  Add an Elastic Inference accelerator  
Additional charges apply.

[Cancel](#) [Previous](#) **Review and Launch** [Next: Add Storage](#)

# Assigning Instance Details – HDD Size

1. Choose AMI    2. Choose Instance Type    3. Configure Instance    4. Add Storage    5. Add Tags    6. Configure Security Group    7. Review

## Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-0c14c41956df80e73	12	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

- 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.

# Assigning Instance Details – Security Group

1. Choose AMI    2. Choose Instance Type    3. Configure Instance    4. Add Storage    5. Add Tags    6. Configure Security Group    7. Review

## Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:  Create a new security group

Select an existing security group



Security Group ID	Name	Description	Actions
<input type="checkbox"/> sg-e26d53d1	default	default VPC security group	<a href="#">Copy to new</a>
<input checked="" type="checkbox"/> sg-095b8df85e4b793c7	Jenkins Server Security Group	Jenkins Server Security Group	<a href="#">Copy to new</a>

Inbound rules for sg-095b8df85e4b793c7 (Selected security groups: sg-095b8df85e4b793c7)

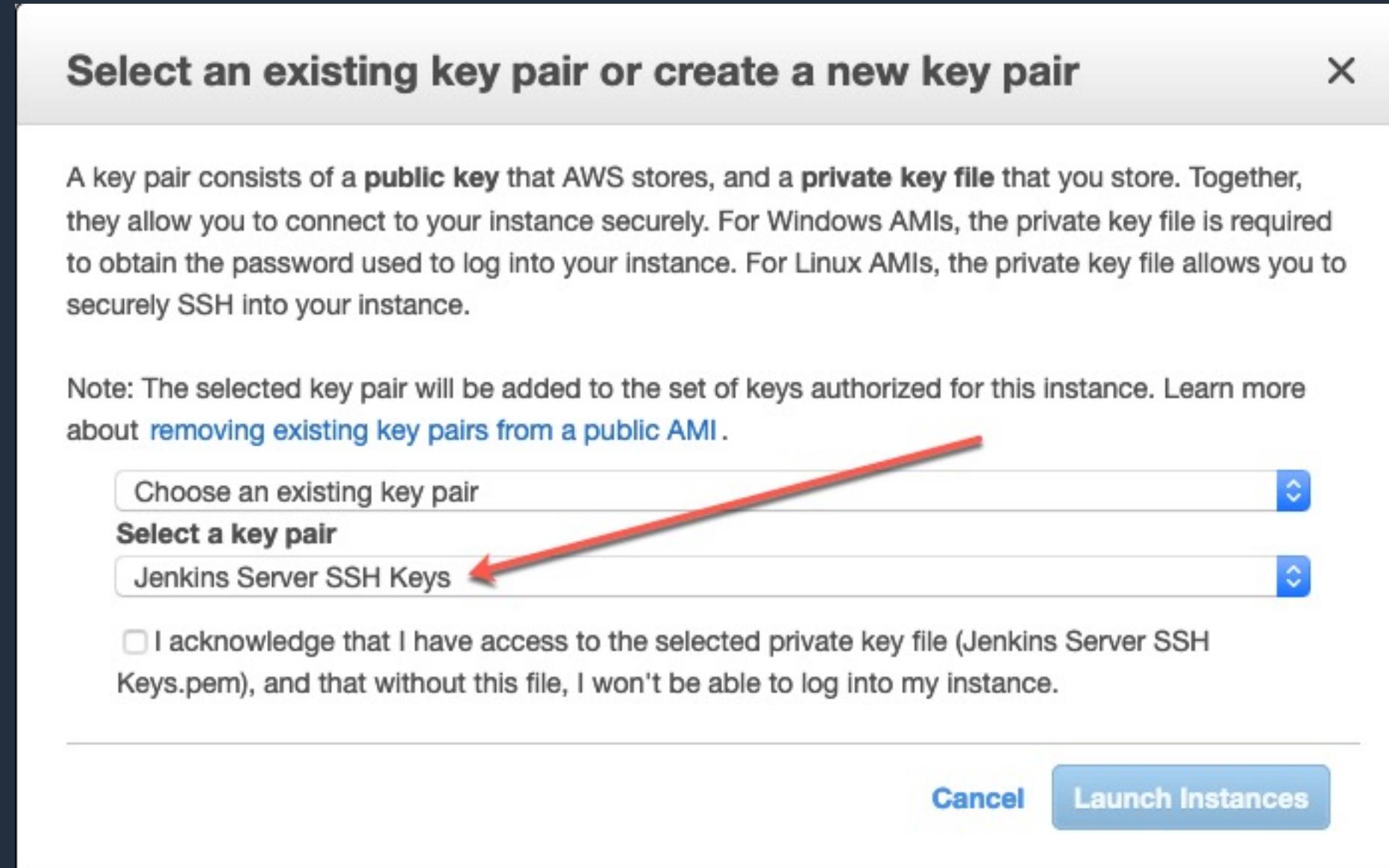


Type <i>i</i>	Protocol <i>i</i>	Port Range <i>i</i>	Source <i>i</i>	Description <i>i</i>
HTTP	TCP	80	100.7.85.251/32	Jenkins Inbound HT...
Custom TCP Rule	TCP	8080	100.7.85.251/32	Jenkins Inbound TC...
SSH	TCP	22	100.7.85.251/32	Jenkins Inbound SS...

[Cancel](#) [Previous](#) [Review and Launch](#)

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

# Assigning Instance Details – Assign a Security Key Pair



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

# Determining your Public IP

EC2 > Instances > i-02a217c43710a02e5

Instance summary for i-02a217c43710a02e5 [Info](#)

Updated less than a minute ago

C Connect Instance state ▾

Instance ID	<b>Public IPv4 address</b> <a href="#">54.202.101.104   open address</a>	Private IPv4 addresses <a href="#">172.31.11.126</a>
Instance state	Public IPv4 DNS <a href="#">ec2-54-202-101-104.us-west-2.compute.amazonaws.com   open address</a>	Private IPv4 DNS <a href="#">ip-172-31-11-126.us-west-2.compute.internal</a>
Instance type	Elastic IP addresses -	VPC ID <a href="#">vpc-c17f4cb9</a>
AWS Compute Optimizer finding	IAM Role -	Subnet ID <a href="#">subnet-b6a639eb</a>
<a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a> <a href="#">Learn more</a>		

A red arrow points from the text "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." to the highlighted Public IPv4 address field.

- 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.

# 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*

# 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
jenkins
jenkins/primary_db
No packages marked for update
[ec2-user@ip-172-31-11-126 ~]$ | 3.7 kB 00:00:00
| 2.9 kB 00:00:00
| 37 kB 00:00:00
```

```
[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)
```

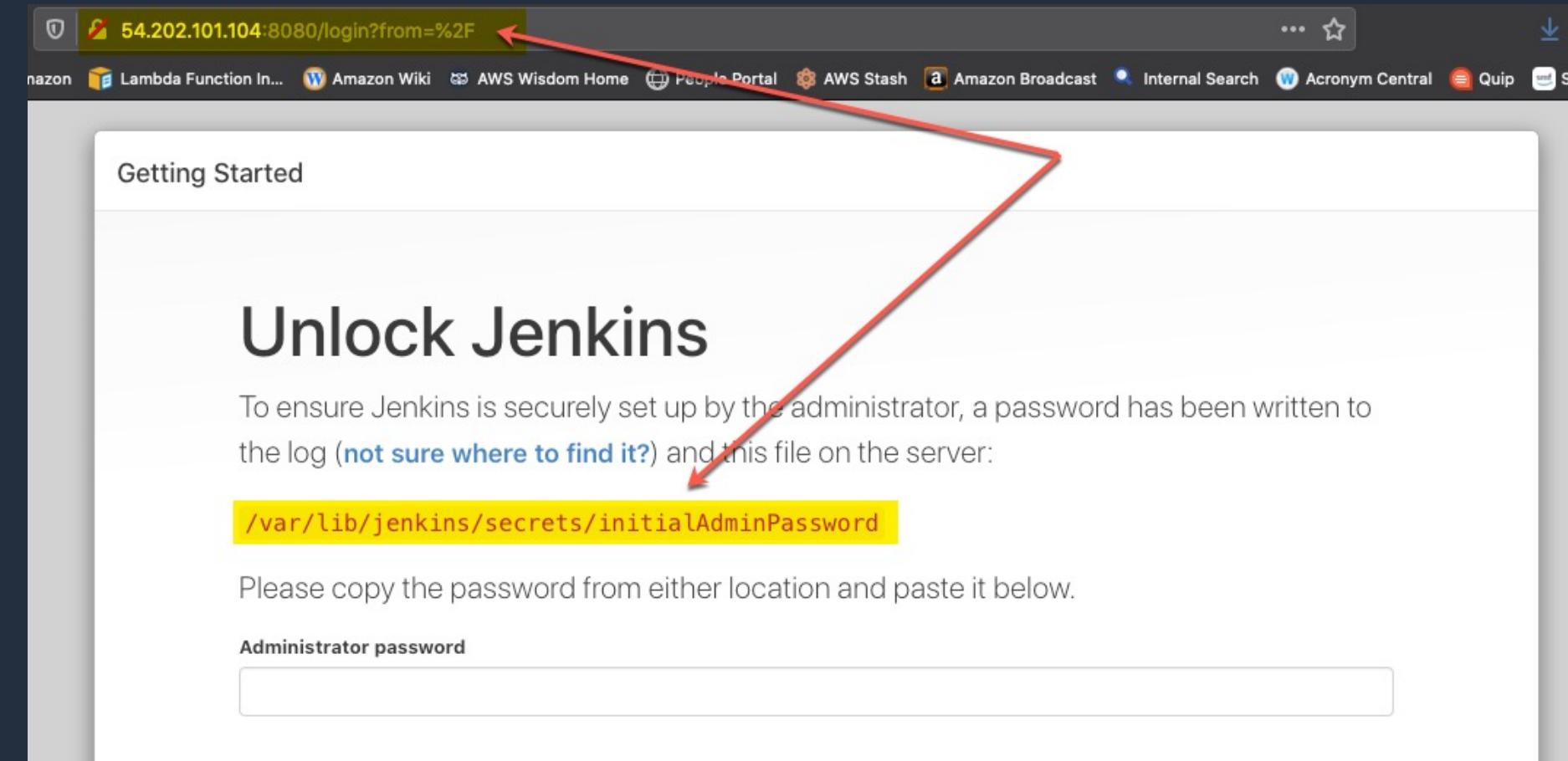
# Update AWS CLI on your EC2 Instance

- 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      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 ~]$
```

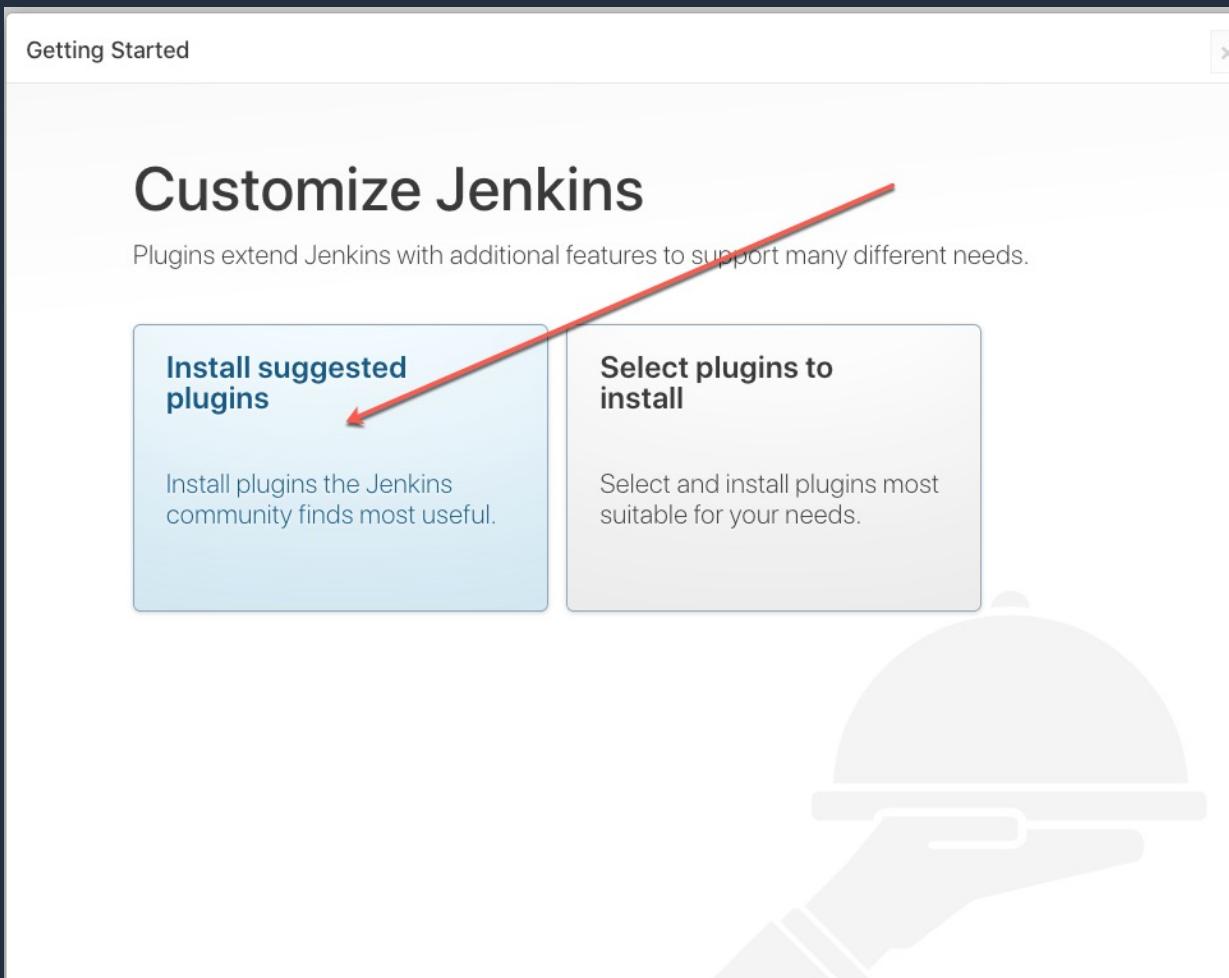
# Browse to Jenkins and initial login

- 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 ~]$
```

# Setup Suggested Plugins



Getting Started

## Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	⌚ Workspace Cleanup	⌚ Ant	⌚ Gradle
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline: Stage View
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication
⌚ LDAP	⌚ Email Extension	⌚ Mailer	

\*\* SSH server  
Folders  
\*\* Trilead API  
OWASP Markup Formatter  
\*\* Structs  
\*\* Pipeline: Step API  
\*\* Token Macro  
Build Timeout  
\*\* Credentials  
\*\* Plain Credentials  
\*\* SSH Credentials  
Credentials Binding  
\*\* SCM API  
\*\* Pipeline: API  
Timestamper  
\*\* Caffeine API  
\*\* Script Security  
\*\* Plugin Utilities API  
\*\* Font Awesome API  
\*\* Popper.js API  
\*\* JQuery3 API

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

# Create the Admin User Account

Getting Started

## Create First Admin User

Username:  

Password:  

Confirm password:  

Full name:

E-mail address:

Jenkins 2.289.1

[Skip and continue as admin](#)  [Save and Continue](#)

- Additional Administrator accounts can be configured after setup is completed

# Save the Instance Configuration URL

Getting Started

## Instance Configuration

Jenkins URL:  

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.289.1 Not now  Save and Finish

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

# Congrats!

Getting Started

## Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)



Jenkins 2.289.1

- Jenkins installation and initial configuration is complete!

# Jenkins Configuration

Basic Plugins and Configuration Settings



# 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:

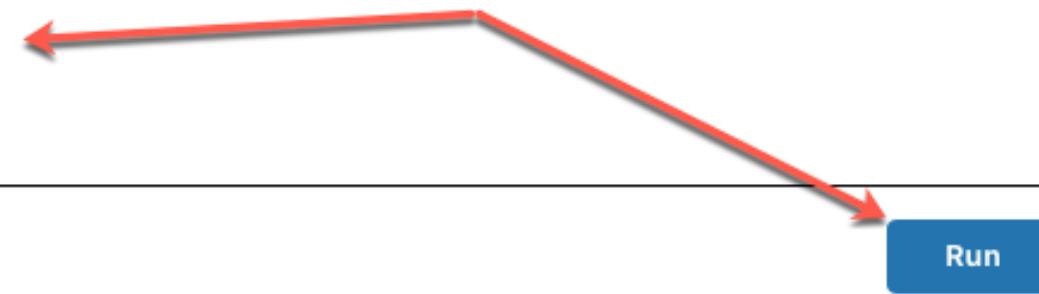
## Script Console

Type in an arbitrary [Groovy script](#) and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use `System.out`, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

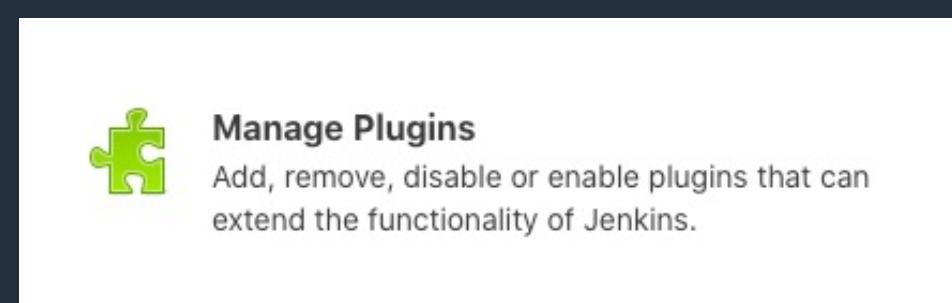
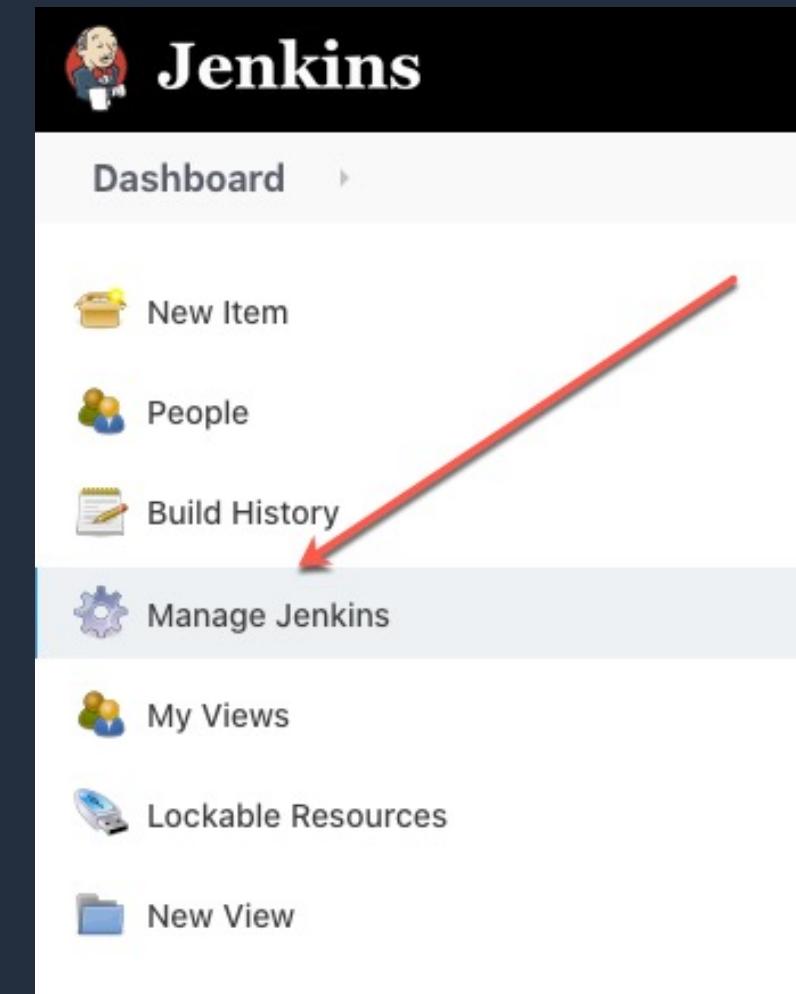
All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported.

```
1 Jenkins.instance.pluginManager.plugins.each{
2   plugin ->
3     println ("${plugin.getDisplayName()} (${plugin.getShortName()}): ${plugin.getVersion()}")
4 }
```



# 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



# Jenkins Plugins – Installation

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

The screenshot shows the Jenkins Plugin Manager interface. At the top, there is a search bar with the text "amazon web services". Below the search bar is a navigation bar with tabs: "Updates" (selected), "Available", "Installed", and "Advanced". On the left side, there is a sidebar with links: "Back to Dashboard", "Manage Jenkins", and "Update Center". The main content area displays a list of available Jenkins plugins:

Name	Version	Released
Amazon Web Services SDK api-plugin aws	1.11.995	2 mo 3 days ago
AWS SQS Build Trigger Build Triggers	2.0.1	3 yr 5 mo ago
This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.		
AWS CodeCommit Trigger aws External Site/Tool Integrations Source Code Management related Build Triggers	3.0.5	8 mo 6 days ago
This plugin triggers builds on all events received via Amazon Web Services Simple Queue Service (AWS SQS) on a specified Queue.		
AWS SQS Build Trigger External Site/Tool Integrations Source Code Management related Build Triggers	1.4	4 yr 11 mo ago
This plugin triggers builds on events from CodeCommit that are published via Amazon Web Services Simple Queue Service (AWS SQS).		

At the bottom of the page, there are three buttons: "Install without restart", "Download now and install after restart", and "Check now". A red arrow points from the search bar to the "Available" tab, and another red arrow points from the "Available" tab to the "Install without restart" button.

# 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

The screenshot shows the Jenkins CLI interface. At the top left is the Jenkins logo and navigation links: Dashboard, Jenkins CLI (highlighted with a red arrow), New Item, People, Build History, Manage Jenkins (highlighted with a red arrow), Convert To Pipeline, and My Views. The main content area has a title 'Jenkins CLI' with a subtitle: 'You can access various features in Jenkins through a command-line tool. See [the documentation](#) for more details of this feature.' Below this is a terminal window displaying the command: 'java -jar jenkins-cli.jar -s http://34.209.223.199:8080/ -webSocket help'. A red arrow points to this command. At the bottom is a section titled 'Available Commands' listing three commands: 'add-job-to-view' (Adds jobs to view.), 'build' (Builds a job, and optionally waits until its completion.), and 'cancel-quiet-down' (Cancel the effect of the "quiet-down" command.).

- 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 blueocean-pipeline-editor blueocean-bitbucket-pipeline blueocean-autofavorite blueocean-dashboard blueocean-commons blueocean-config blueocean-display-url blueocean-events blueocean-git-pipeline blueocean-github-pipeline cloudbees-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 -deploy -restart
```

# 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

### Extended E-mail Notification

SMTP server	smtp.office365.com
SMTP Port	587

### Slack

Workspace	<input type="text"/>
Credential	<select>- none -</select> <a href="#">Add</a>
Default channel / member id	<input type="text"/>
Custom slack app bot user	<input checked="" type="checkbox"/>

The screenshot shows the Jenkins Global Configuration page. On the left, there's a sidebar with links: Dashboard, New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins (which has a red arrow pointing to it), and Convert To Pipeline. Below the sidebar, there's a "System Configuration" section with a "Configure System" link and a description: "Configure global settings and paths." At the bottom right, the AWS logo is visible.

# 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

## Create User

Username:

Password:

Confirm password:

Full name:

E-mail address:

**Create User**

### Password

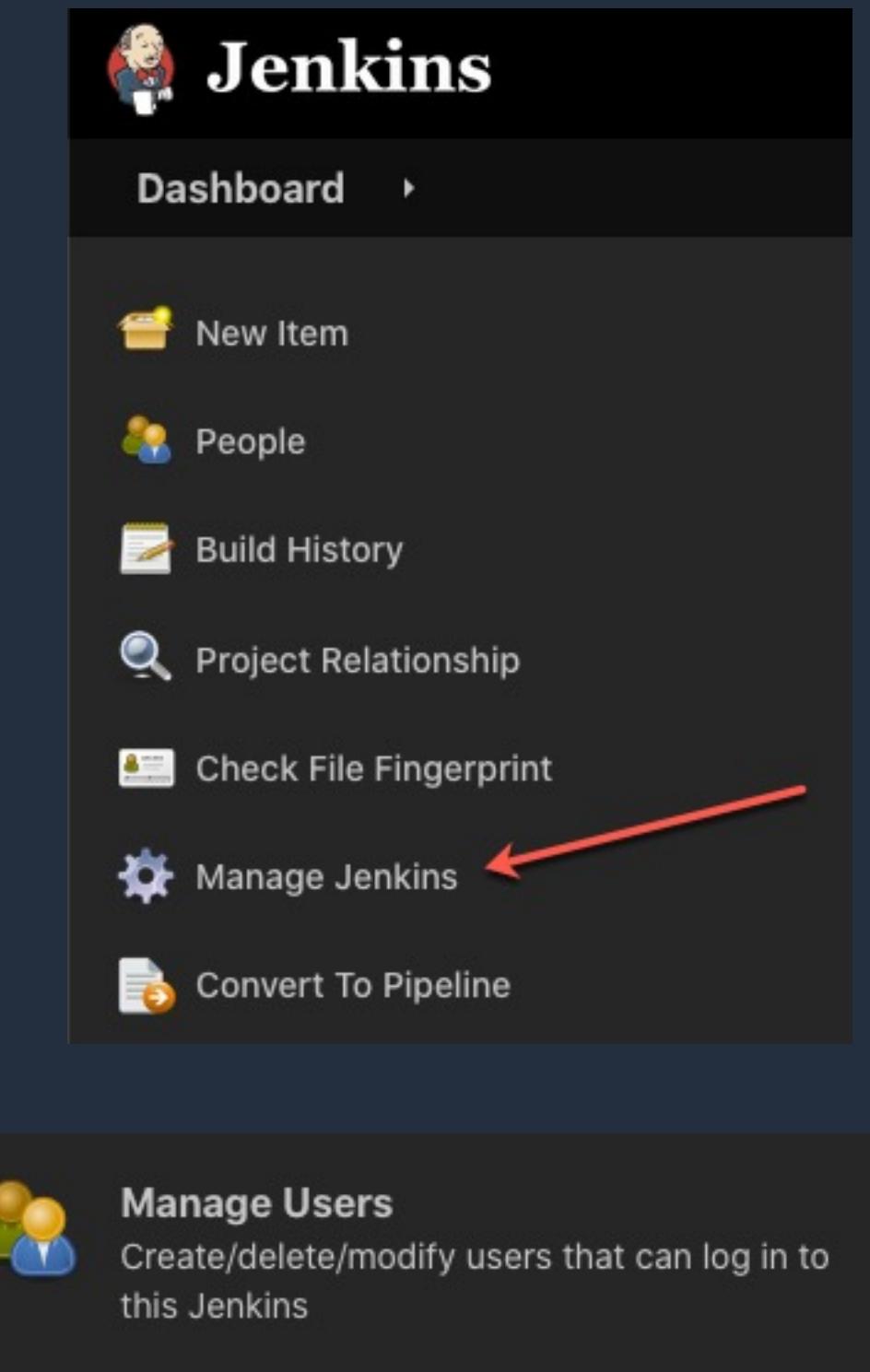
Password:

Confirm Password:

### Slack User Settings

User Id

Disable Notifications



# 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

## Stores scoped to Jenkins

The screenshot shows the Jenkins Global Credentials interface. At the top, there's a navigation bar with 'P' and 'Store ↓'. Below it, a sidebar lists 'Jenkins' and other domains. The main area is titled 'Domains' and shows a 'global' domain with a 'Add credentials' button highlighted by a mouse cursor.

The screenshot shows the Jenkins Global credentials screen. The navigation path is 'Dashboard > Credentials > System > Global credentials (unrestricted)'. On the left, there's a 'Back to credential domains' link and an 'Add Credentials' button. A dropdown menu titled 'Kind' is open, listing various credential types. The 'AWS Credentials' option is highlighted with a blue selection bar and an arrow pointing to it from the bottom-left.

The screenshot shows the Jenkins sidebar. It includes links for 'Dashboard', 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins' (which is highlighted with a red arrow), and 'Convert To Pipeline'. At the bottom, there's a 'Manage Credentials' section with a 'Configure credentials' link.

# 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

Jenkins

search log out

Alex Hannah

Dashboard > Credentials > System > Global credentials (unrestricted) >

Add Credentials

Kind: AWS Credentials

Scope: Global (Jenkins, nodes, items, all child items, etc)

ID: { If you leave this blank, Jenkins generates a GUID automatically }

Description: { Description shows up in the Jenkins GUI }

Access Key ID: { AWS Access Key Goes Here }

Secret Access Key: { AWS Secret Access Key Goes Here }

IAM Role Support

Advanced...

OK

The screenshot shows the Jenkins Global credentials (unrestricted) page. It lists two entries under the 'Kind' column: 'AWS Credentials'. The first entry has an ID of '1e5dab5a-27a2-4660-b960-7553aa5154be' and a name of 'AKIA4P7BNWNCXHJRHCIGI (AWS AH - Jenkins API User)'. The second entry has an ID of '71b568ab-3ca8-4178-b03f-c112f0fd5030' and a name of 'AKIAVGUF7PUXFINJEPNL (AWS RS - Jenkins API User)'. Both entries are of type 'AWS Credentials'.

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
            }
        }
    }
}
```

The screenshot shows a Jenkins Pipeline script. It includes a 'stage' block named 'Create an Amazon Connect Instance'. Inside the stage, there is a 'steps' block. The first step is an 'echo' command. The second step is a 'withAWS' block. The 'withAWS' block takes three parameters: 'credentials' set to '71b568ab-3ca8-4178-b03f-c112f0fd5030', 'region' set to 'us-east-1', and an anonymous block. The anonymous block contains a 'script' block with several AWS commands and variable assignments.

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
1e5dab5a-27a2-4660-b960-7553aa5154be	AKIA4P7BNWNCXHJRHCIGI (AWS AH - Jenkins API User)	AWS Credentials	AWS AH - Jenkins API User
71b568ab-3ca8-4178-b03f-c112f0fd5030	AKIAVGUF7PUXFINJEPNL (AWS RS - Jenkins API User)	AWS Credentials	AWS RS - Jenkins API User
ec-user-jenkins	ec2-user	SSH Username with private key	

# Jenkins Jobs

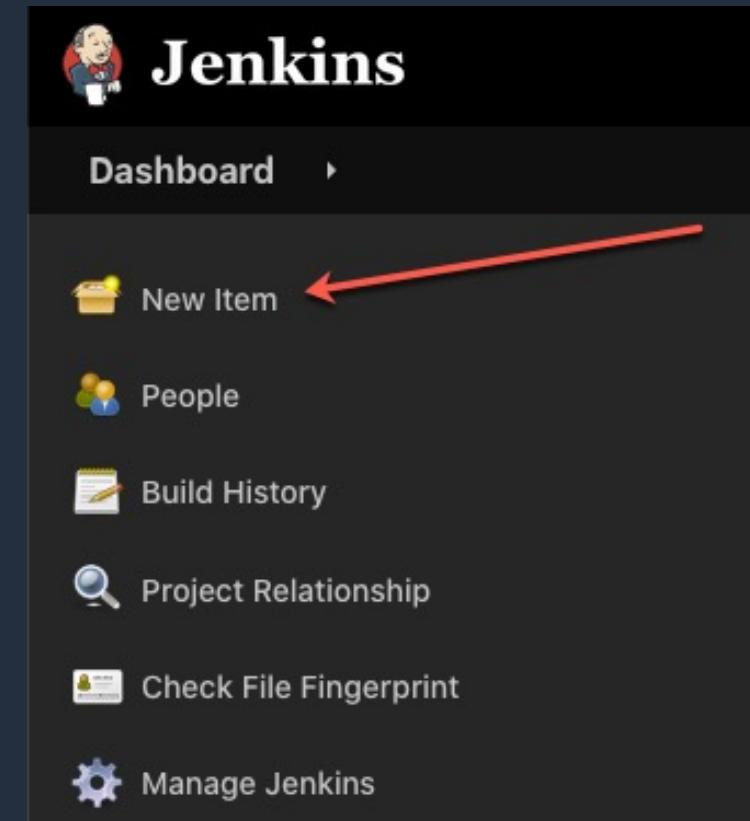
Pipeline Jobs for Amazon Connect



# 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*



## Enter an item name

Demo Pipeline » 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.

**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.

# 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

The screenshot shows the Jenkins Pipeline Job Options interface. At the top, there are four tabs: General (highlighted in yellow), Build Triggers, Advanced Project Options, and Pipeline. A red arrow points from the text "Advanced Project Options" in the list above to the "Advanced Project Options" tab here. Below the tabs is a "Description" field containing "A Demo Pipeline in Jenkins". Underneath the description is a link "[Plain text] Preview". At the bottom right are two buttons: "Save" (blue) and "Apply" (white).

Save = Page Reload  
Apply = Stay on Page

## General Setting of Interest

This screenshot shows the "General" settings section of the Jenkins Pipeline Job Options. It includes several checkboxes:

- Discard old builds
- Do not allow concurrent builds
- Do not allow the pipeline to resume if the controller restarts
- GitHub project
- Pipeline speed/durability override
- Preserve stashes from completed builds
- This project is parameterized
- Throttle builds

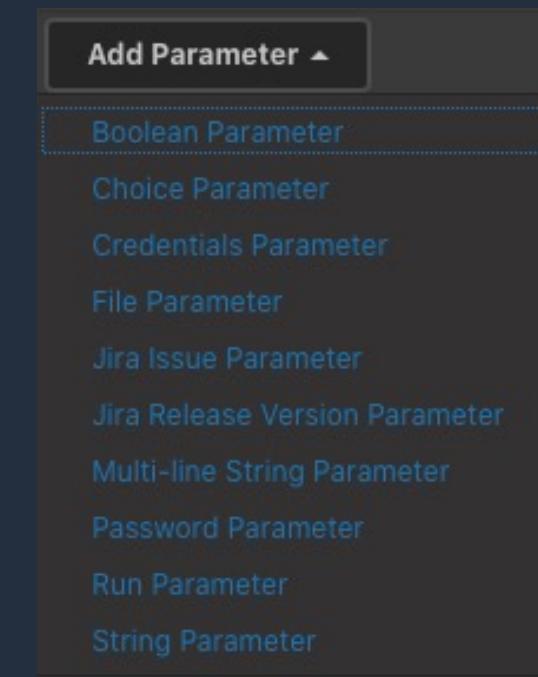
## Build Trigger Setting of Interest

This screenshot shows the "Build Triggers" section of the Jenkins Pipeline Job Options. It includes several checkboxes:

- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM
- Disable this project
- Quiet period
- Trigger builds remotely (e.g., from scripts)

# 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!

This project is parameterized

**String Parameter**

Name ?

Instance\_Alias

Default Value ?

Description ?

Please enter the Instance Alias you wish to use, this is the same thing as the Instance Name

[Plain text] [Preview](#)

Trim the string ?

## Pipeline RS - Connect - Create Instance - Manual Params

This build requires parameters:

Instance\_Alias

Please enter the Instance Alias you wish to use, this is the same thing as the Instance Name

Enable\_Inbound\_Calls

Leave this checkbox enabled if you wish to accept inbound calls to this instance.

Enable\_Outbound\_Calls

Leave this checkbox enabled if you wish to place outbound calls with this instance.

Identity\_Management\_Type

CONNECT\_MANAGED ?

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

## Definition

### Pipeline script

#### Script

```
1 import groovy.json.JsonSlurper
2
3
4 @NonCPS
5 def jsonParse(def json) {
6     new groovy.json.JsonSlurper().parseText(json)
7 }
8
9 def ARN
10 pipeline {
11     agent any
12     stages {
13         stage('Initialization') {
14             steps {
15                 echo 'Starting the Script meow'
16             }
17         }
18         stage('Create an Amazon Connect Instance'){
19             steps {
20                 echo 'Creating the Amazon Connect Instance'
21                 withAWS(credentials: '71b568ab-3ca8-4178-b03f-c112f0fd5030', region: 'us-east-1') {
22
 Use Groovy Sandbox


### Pipeline Syntax


```

```
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  
    }  
}
```

# Jenkins Job Output ( Stages and Logs )

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

## Pipeline RS - Connect - Create Instance - APIs

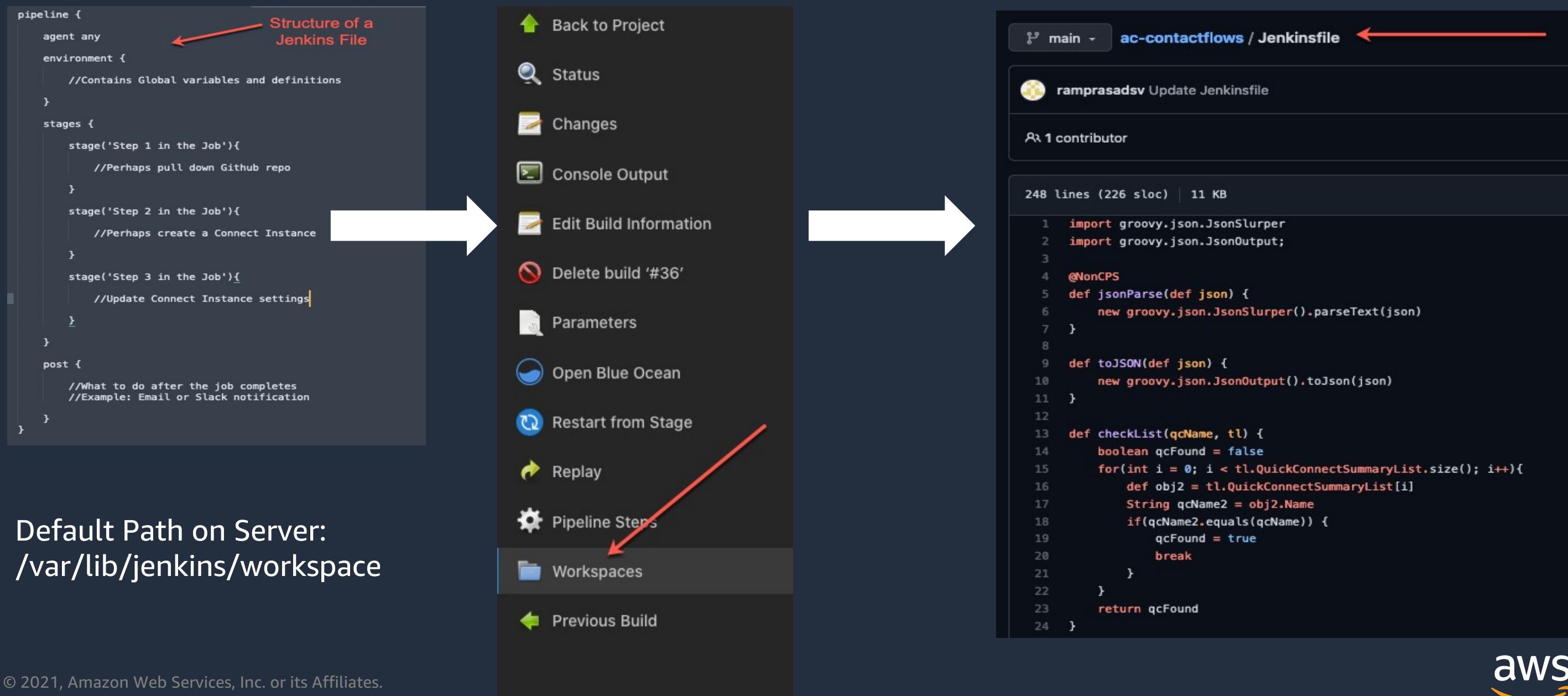


## Permalinks

- Last build (#39), 3 days 4 hr ago
- Last stable build (#39), 3 days 4 hr ago
- Last successful build (#39), 3 days 4 hr ago
- Last failed build (#38), 3 days 4 hr ago
- Last unsuccessful build (#38), 3 days 4 hr ago
- Last completed build (#39), 3 days 4 hr ago

# 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



# 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

### Build Triggers

- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM
- Disable this project
- Quiet period
- Trigger builds remotely (e.g., from scripts)

### Advanced Project Options

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/r...  
[REDACTED]

Credentials

- none -

Branches to build

Branch Specifier (blank for 'any')

\*/main

Repository browser

(Auto)

Additional Behaviours

Add

Script Path

Jenkinsfile

Lightweight checkout

Pipeline Syntax

Save  Apply

# 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

The screenshot shows two side-by-side views. On the left is a sidebar menu for GitHub settings, with 'Webhooks' highlighted by a red arrow. On the right is a 'Webhooks / Manage webhook' configuration page. It contains a descriptive text about sending POST requests to a payload URL, a 'Payload URL \*' input field containing 'http://3.236.244.143:8080/github-webhook/' (also highlighted by a red arrow), a 'Content type' dropdown set to 'application/json', and a 'Secret' section with a note about changing it. A large green box at the bottom right of the config page contains the text: 'If you've lost or forgotten this secret, you can change it, but be aware that any integrations using this secret will need to be updated. — [Change Secret](#)'.

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Actions

Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL \*

http://3.236.244.143:8080/github-webhook/

Content type

application/json

Secret

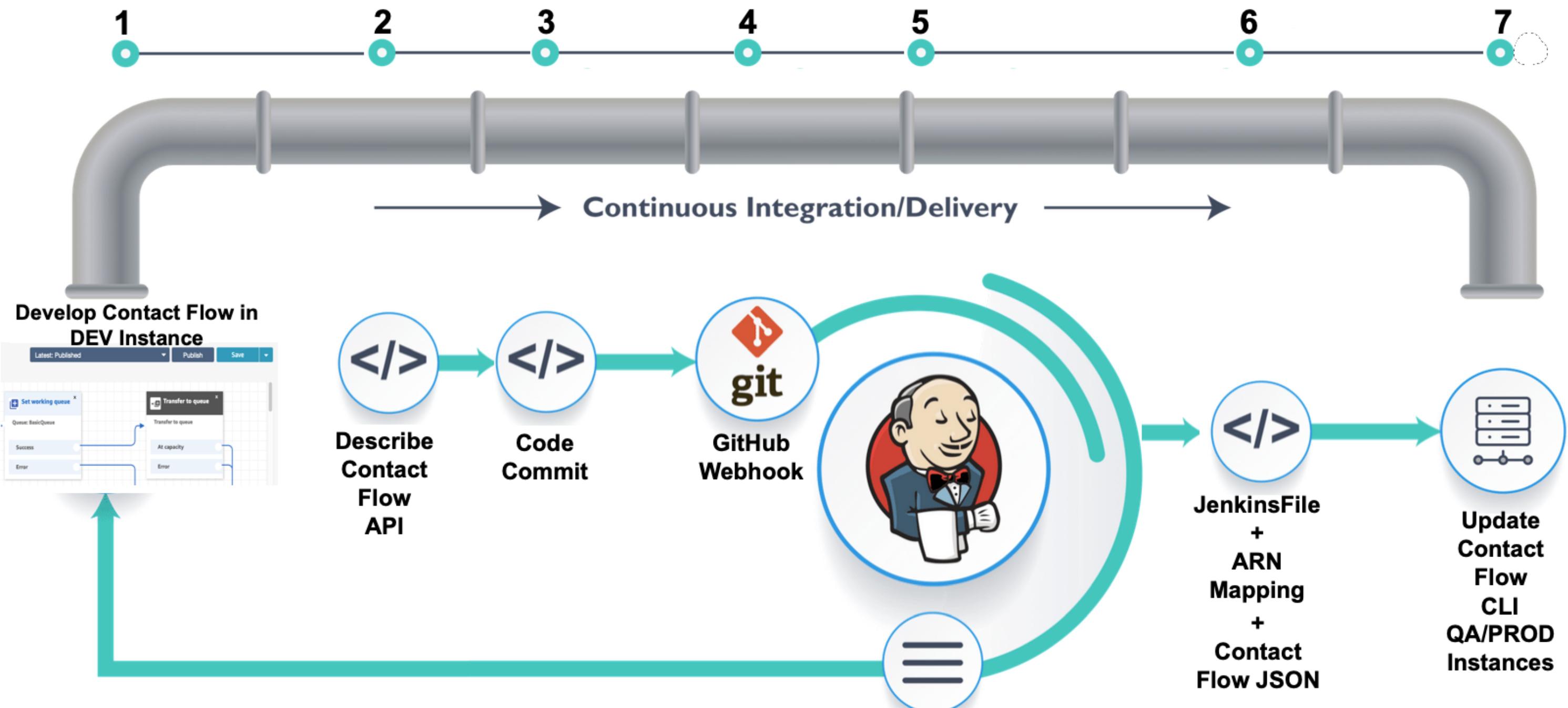
If you've lost or forgotten this secret, you can change it, but be aware that any integrations using this secret will need to be updated. — [Change Secret](#)

# Amazon Connect – CI/CD

Understanding the CI/CD Workflow

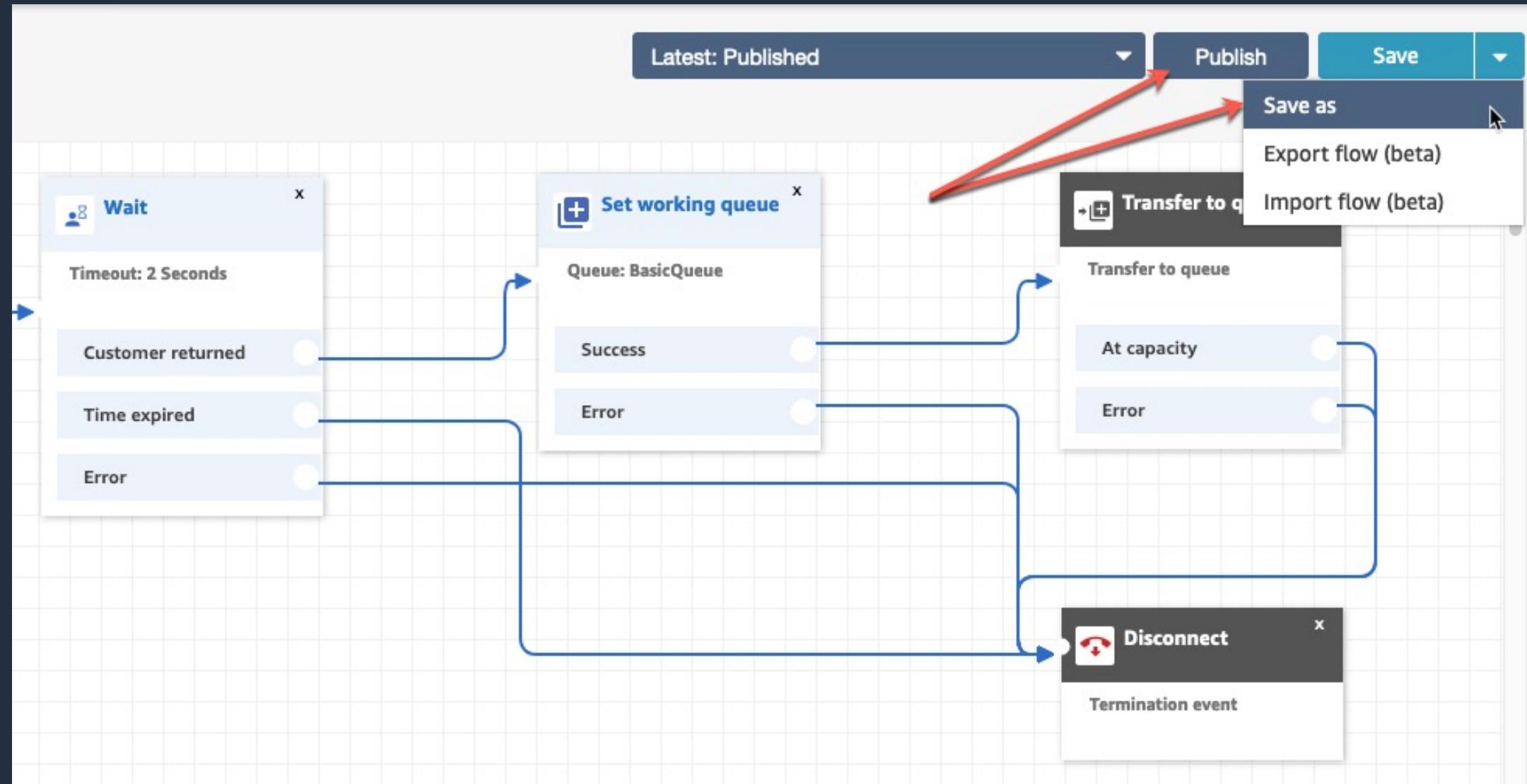


# Amazon Connect CI/CD Workflow



# 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! )

The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** `https://{{service}}.{{region}}.{{awsurl}}/contact-flows/{{instanceID}}/{{contactFlowID}}`
- Headers:** (8)
- Body:** (Pretty) JSON response (highlighted in yellow). The response is a JSON object representing a Contact Flow, with several fields highlighted by red arrows:
  - `"Arn": "arn:aws:connect:us-east-1:858929345349:instance/8e56a433-8801-49b0-a825-144f24a682e3/contact-flow/7cd6f003-87e5-497e-ab67-b29e4f8241c7"`
  - `"Content": "..."` (The entire Content field is highlighted in yellow)
  - `"Tags": {}`
- Status:** 200 OK
- Time:** 205 ms
- Size:** 3.62 KB

- [https://docs.aws.amazon.com/connect/latest/APIReference/API\\_DescribeContactFlow.html](https://docs.aws.amazon.com/connect/latest/APIReference/API_DescribeContactFlow.html)

# 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

The screenshot shows a GitHub commit dialog over a repository page. The repository name is 'ac-flows-via-github'. The commit message is 'Updated JenkinsFile 61421' followed by 'Updated the JenkinsFile for Amazon Connect Contact Flows 6-14-21'. The dialog offers two options: 'Commit directly to the main branch.' (selected) and 'Create a new branch for this commit and start a pull request. Learn more about pull requests.' A red arrow points to the second option.

main • 1 branch • 0 tags

Go to file Add file • Code

ramprasadsdv Update Jenkinsfile bb7dada 5 days ago 8 commits

Jenkinsfile Update Jenkinsfile

LICENSE Initial commit

README.md Initial commit

a-test1.json Create a-test1.json

arnmapping.json Create arnmapping.json

parameters.json Update parameters.json

README.md

Commit changes

Updated JenkinsFile 61421

Updated the JenkinsFile for Amazon Connect Contact Flows 6-14-21

• -o- Commit directly to the main branch.

• ↗ Create a new branch for this commit and start a pull request. Learn more about pull requests.

Commit changes Cancel

This example will demonstrate of how the solution will pick the flow from Github and publish to a different instance

# 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



## Build Triggers

- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM
- Disable this project
- Quiet period
- Trigger builds remotely (e.g., from scripts)



## Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

`https://github.com/ramprasadsdv/ac-flows-via-github.git`

Credentials

- none -

Branches to build

Branch Specifier (blank for 'any')

`*/main`

# 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



# 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>



# 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>



# 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>



# GitHub Repo's

Ram's GitHub Repo:

<https://github.com/ramprasadsyv>

Alex's GitHub Repo:

<https://github.com/Rodeorat316>



# Q&A



# Thank you!

