
Linux Academy
Amazon Web Services
What is CodeDeploy?

## What is CodeDeploy?

- CodeDeploy is an AWS service for automating the deployment process of your applications, from Git-based version control systems or S3 buckets to Amazon EC2 instances, on-premise instances, or both.

- ***Automation:***
  - You can easily automate your code deployment to development, test, and production environments.
- ***Scale:***
  - Deploy your code to one or thousands of instances at once.
- ***Reduced Downtime:***
  - Rolling updates allow you to decrease downtime by allowing you to track application heath and stop/rollback deployments if there are errors.
- ***Control:***
  - Easily keep track of your deployments by receiving reports that list when and where each of your application revisions is deployed.
- ***Platform-agnostic:***
  - CodeDeploy is built to work with any application.

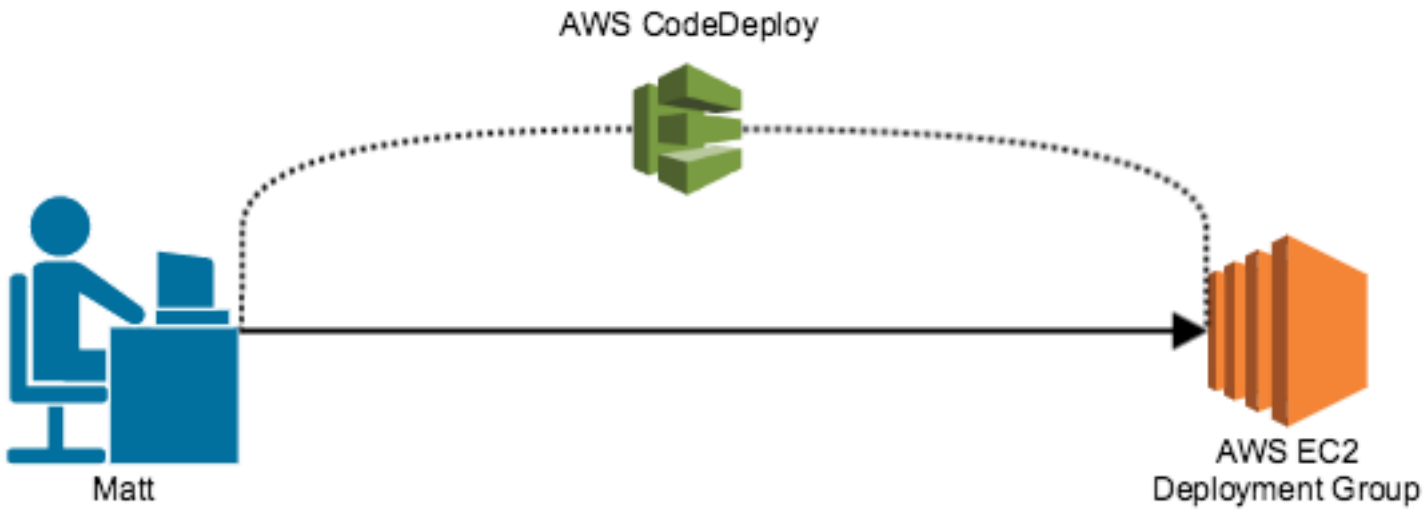
# CodeDeploy: Workflow

# CodeDeploy: Workflow

Matt

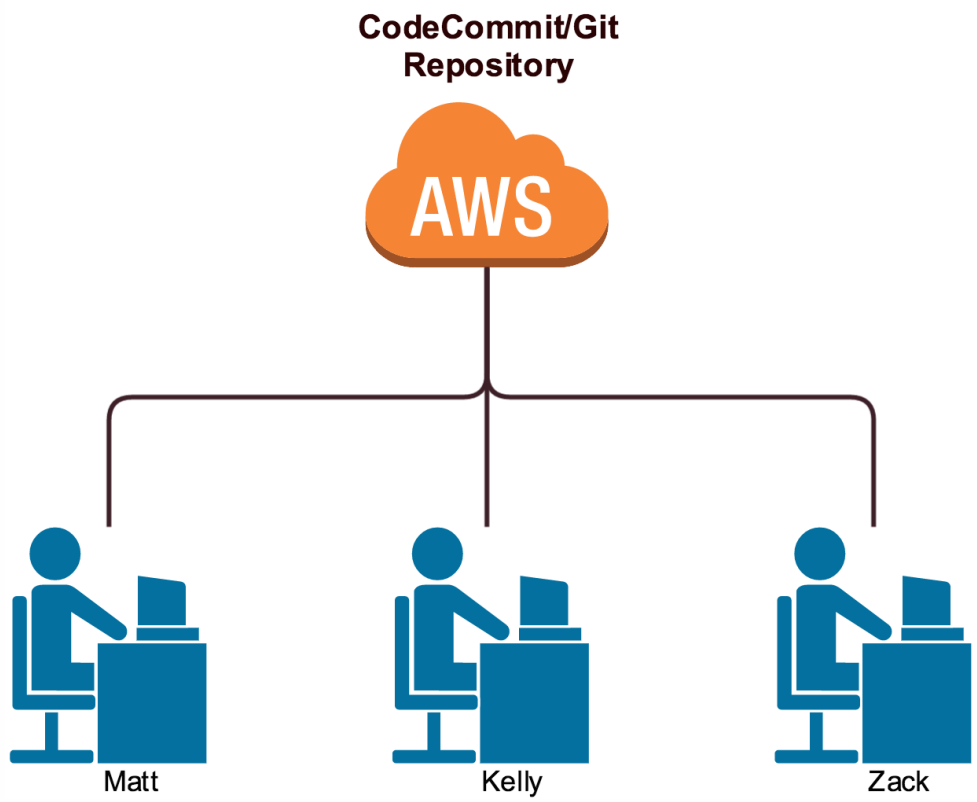
AWS EC2
Deployment Group

# CodeDeploy: Workflow



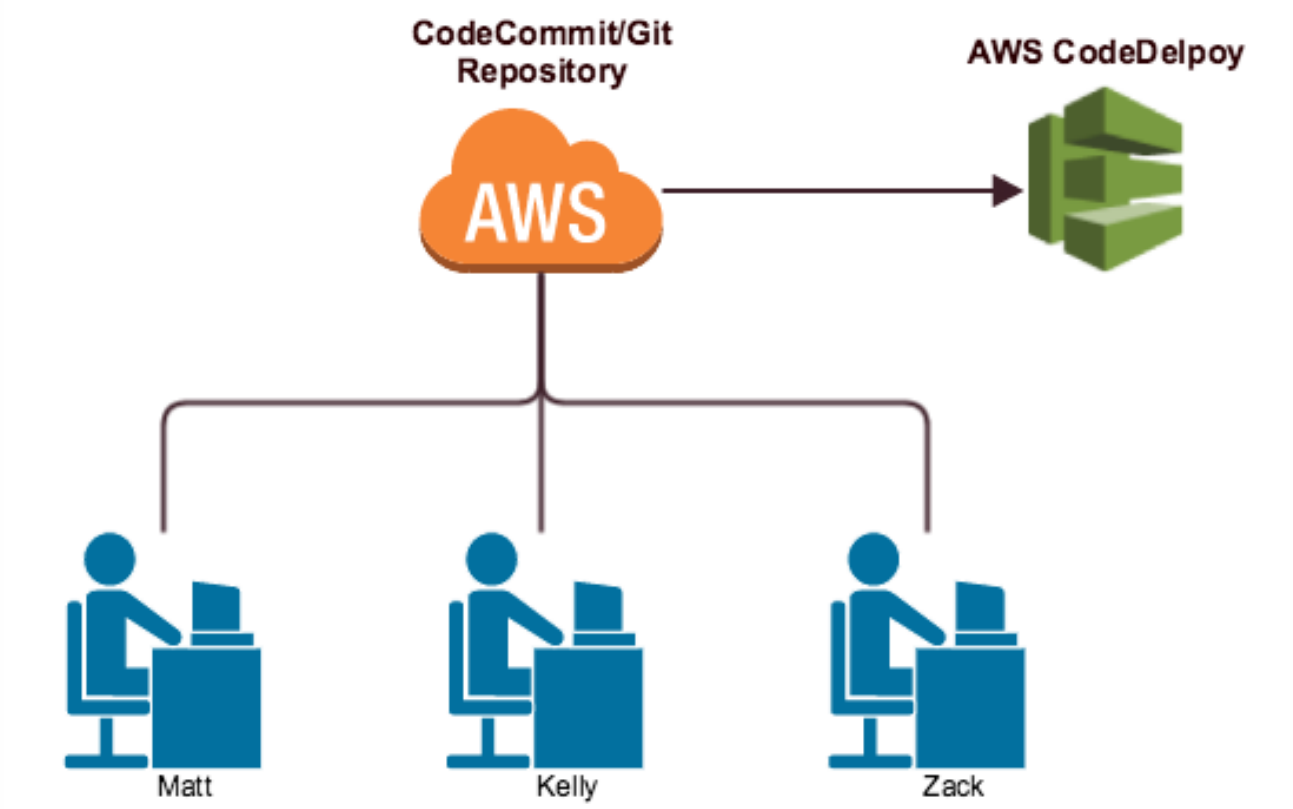
Linux Academy

Meet our ***WonderWidgets*** Development Team



Linux Academy

Meet our ***WonderWidgets*** Development Team




Linux Academy
Amazon Web Services
What is CodeDeploy?
Thank you for watching!


# Amazon Web Services

## CodeDeploy: Setup & Configuration

# CodeDeploy: Setup & Configuration

1) Provision an IAM user with a custom CodeDeploy Policy
   - *Gives a non-admin user the rights to manage all the elements needed to use CodeDeploy.*

2) Create an Instance Profile
   - *This allows you to launch EC2 instances that are configured for use with CodeDeploy.*

3) Create a Service Role
   - *This will allow CodeDeploy to communicate and interact with other AWS Services.*

4) Install the AWS Command Line Interface (CLI)
   - *For Windows: (2) Windows: Git & AWS CLI Installation*
   - *For OSX/Linux: (7) OSX/Linux: AWS CLI Installation*

******These videos are located under the CodeCommit section of this course***

Linux Academy

# Amazon Web Services

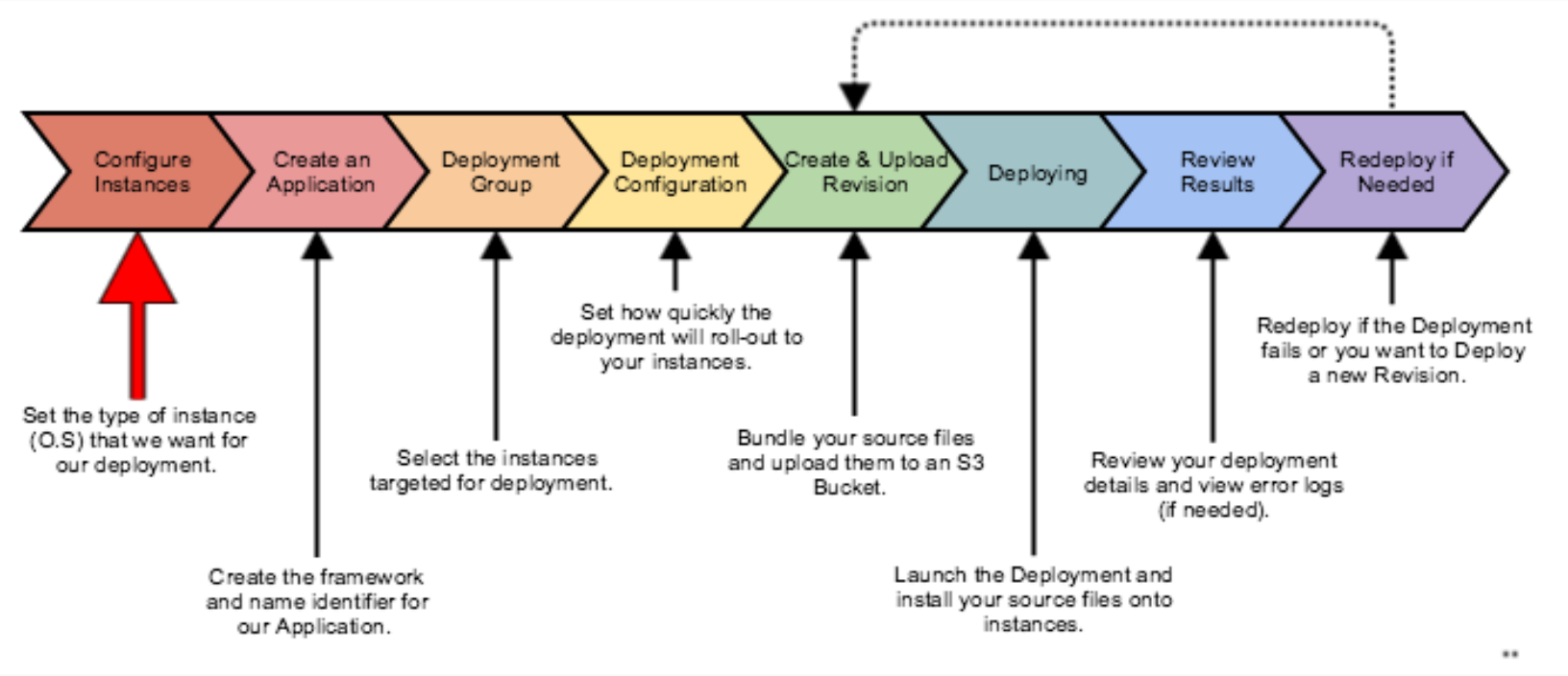
## CodeDeploy: Setup & Configuration

# Thank you for watching!


# Amazon Web Services

## CodeDeploy: Configuring Instances

# CodeDeploy Workflow:



# Setting up Deployment Instances

- What type of instances does CodeDeploy support?
  - *Amazon Linux*
  - *Ubuntu Server*
  - *Red Hat Enterprise Linux (RHEL)*
  - *Windows Server*

- What methods can we use to launch & configure instances?
  - EC2 Instances:
    - *Manually create & configure in the AWS Console or CLI*
    - *CloudFormation template*
  - On-Premise:

- For the purpose of this lesson, we are going to review how to manually set up and configure an EC2 instance for CodeDeploy via the AWS console.

- If you are interested in learning about creating instances for CodeDeploy via CloudFomation, watch the video section in our *AWS Certified Developer Course titled "Amazon CloudFormation"*.

# Configuration Steps:

1) Launch a new Amazon Linux AMI
2) Select appropriate instance Type
3) Set IAM role to the Instance Profile we created in the Setup & Configuration lesson
4) Open ‘Advanced Details’ and add the following bash script:

```
#!/bin/bash
yum -y update
yum install -y ruby
yum install -y aws-cli
cd /home/ec2-user
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

5) Add storage
6) Add a ‘Name’ Tag to the instance
7) Configure a Security Group
8) Review and Launch
9) Select a key pair
10) Check to see if the AWS CodeDeploy agent has been successfully installed

# AWS CodeDeploy Agent:

- The AWS CodeDeploy agent is a custom software package that must be installed on all instances that will be part of a deployment group. The agent specifies many of the settings that are needed for the instance to interact with CodeDepoy – like directory paths, log files, and deployment polling time intervals. The agent is also customizable, so you can alter it to fit your deployment needs.

- For Amazon, Ubuntu, and RHEL instances:
  - *Name:* *codedeployagent.yml*
  - *Location:* */etc/codedeploy-agent/conf*

- For Window Servers:
  - *Name:* *conf.yml*
  - *Location:* *C:\ProgramData\Amazon\CodeDeploy*

# Configuration Steps:

1) Launch a new Amazon Linux AMI
2) Select appropriate instance Type
3) Set IAM role to the Instance Profile we created in the Setup & Configuration lesson
4) Open ‘Advanced Details’ and add the following bash script:

```
#!/bin/bash
yum -y update
yum install -y ruby
yum install -y aws-cli
cd /home/ec2-user
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

5) Add storage
6) Add a ‘Name’ Tag to the instance
7) Configure a Security Group
8) Review and Launch
9) Select a key pair
10) Check to see if the AWS CodeDeploy agent has been successfully installed

# AWS CodeDeploy Agent:

## *Linux server*

| S3 Bucket Name | Region Name |
|---|---|
| aws-codedeploy-us-east-1 | us-east-1 |
| aws-codedeploy-us-west-1 | us-west-1 |
| aws-codedeploy-us-west-2 | us-west-2 |
| aws-codedeploy-eu-west-1 | eu-west-1 |
| aws-codedeploy-eu-central-1 | eu-central-1 |
| aws-codedeploy-ap-northeast-1 | ap-northeast-1 |
| aws-codedeploy-ap-northeast-2 | ap-northeast-2 |
| aws-codedeploy-ap-southeast-1 | ap-southeast-1 |
| aws-codedeploy-ap-southeast-2 | ap-southeast-2 |
| aws-codedeploy-sa-east-1 | sa-east-1 |

## *Windows server*

| S3 Bucket Name |
|---|
| aws-codedeploy-us-east-1 |
| aws-codedeploy-us-west-1 |
| aws-codedeploy-us-west-2 |
| aws-codedeploy-eu-west-1 |
| aws-codedeploy-eu-central-1 |
| aws-codedeploy-ap-northeast-1 |
| aws-codedeploy-ap-northeast-2 |
| aws-codedeploy-ap-southeast-1 |
| aws-codedeploy-ap-southeast-2 |
| aws-codedeploy-sa-east-1 |

# Configuration Steps:

1) Launch a new Amazon Linux AMI
2) Select appropriate instance type
3) Set IAM role to the Instance Profile we created in the Setup & Configuration lesson
4) Open ‘Advanced Details’ and add the following bash script:

```
#!/bin/bash
yum -y update
yum install -y ruby
yum install -y aws-cli
cd /home/ec2-user
aws s3 cp s3://bucket-name/latest/install . --region region-name
chmod +x ./install
./install auto
```

5) Add storage
6) Add a ‘Name’ Tag to the instance
7) Configure a Security Group
8) Review and Launch
9) Select a key pair
10) Check to see if the AWS CodeDeploy agent has been successfully installed

# CodeDeploy Agent Status:

- Command:
  ***sudo service codedeploy-agent status***

- Good Response:
  ***The AWS CodeDeploy agent is running.***

- Error Responses:
  *Error:* ***No AWS CodeDeploy agent running***
  *Fix:* **sudo service codedeploy-agent start**

  *Error:* ***codedeploy-agent: unrecognized service***
  *Fix:* ***Launch a new instance and double check the bash script for errors, OR double check the permissions policy attached to the Instance Profile (make sure it allows access to S3)***

Linux Academy

# Amazon Web Services

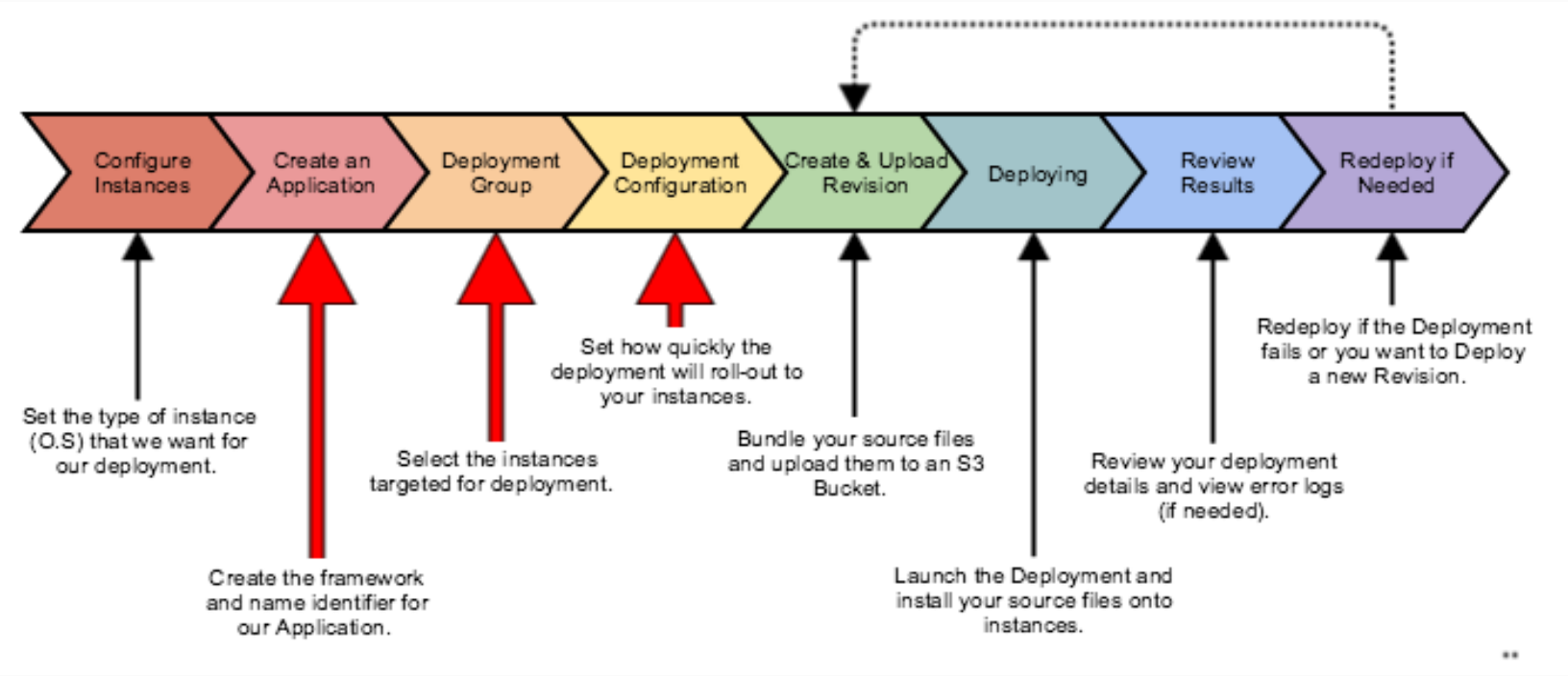
## CodeDeploy: Configuring Instances

# Thank you for watching!


# Amazon Web Services

## CodeDeploy: Creating an Application, Deployment Group, and Deployment Configuration

# CodeDeploy Workflow:



# Understanding the Terminology:

- CodeDeploy **Application**:
  - *The application is simply a name identifier used to reference your deployment settings*

- CodeDeploy **Deployment Group**:
  - *The deployment group is the instance (or instances) you want to target and deploy your code to*

- CodeDeploy **Deployment Configuration**:
  - *Select how many instances in your deployment group your code will deploy to at any given time*

# Managing an Application from the AWS Console:

## Creating an Application, Deployment Group & Configuration:

1) *Navigate to CodeDeploy*
2) *Choose Create a New Application*
3) *Give the Application a name*

***The next set of steps is to set up the Deployment Group within the Application***

4) *Give the deployment group a name*
5) *Select the tag type, Key & value for your instance*

***The next step is to setup the Deployment Configuration within the Application***

6) *Select deployment configuration*
   - *OneAtATime*
   - *AllAtOnce*
   - *HalfAtATime*

***Optional***

7) *Create a trigger*

***Set permissions***

8) *Select the CodeDeploy Service role we create*

9) *Finalize & create the Application*

# Managing an Application from the AWS CLI:

## Creating an Application, Deployment Group & Configuration:

Base CLI Command:

***aws deploy***

1) *Create a new application*

***aws deploy create-application --application-name <NAME>***

2) *Create the deployment group, configuration & other options*

In one command we will specify:

- *Deployment group (tag, key, value)*
- *Deployment Configuration (AllAtOnce, OneAtATime, HalfAtATime)*
- *Trigger (optional)*
- *Service role (permissions)*

***aws deploy create-deployment-group --application-name <NAME> --deployment-group-name <NAME> --ec2-tag-filters Key=Name,Value=<EC2VALUE>,Type=KEY_AND_VALUE --deployment-config-name CodeDeployDefault.<SELECTOPTION> --service-role-arn <SERVICE-ROLE_ARN>***


# Amazon Web Services

## CodeDeploy: Creating an Application, Deployment Group, and Deployment Configuration

# Thank you for watching!


# Amazon Web Services

## CodeDeploy: Editing and Deleting an Application

# Edit & Delete a CodeDeploy Application:

- Changing the Application Name:
  - *AWS Console: N/A*
  - *AWS CLI Commands:*

    ***aws deploy list-applications*** *(list all our applications)*

    ***aws deploy update-application --application-name <NAME> --new-application-name <NEWNAME>***

    *(rename our application)*

- Deleting an Application:
  - AWS CLI Commands:

    **aws deploy delete-application --application-name <NAME>**

    (delete an application)
  - AWS Console:

    *1) Click on the application*

    *2) Scroll to the bottom and click 'Delete Application'*

Linux Academy

# Amazon Web Services

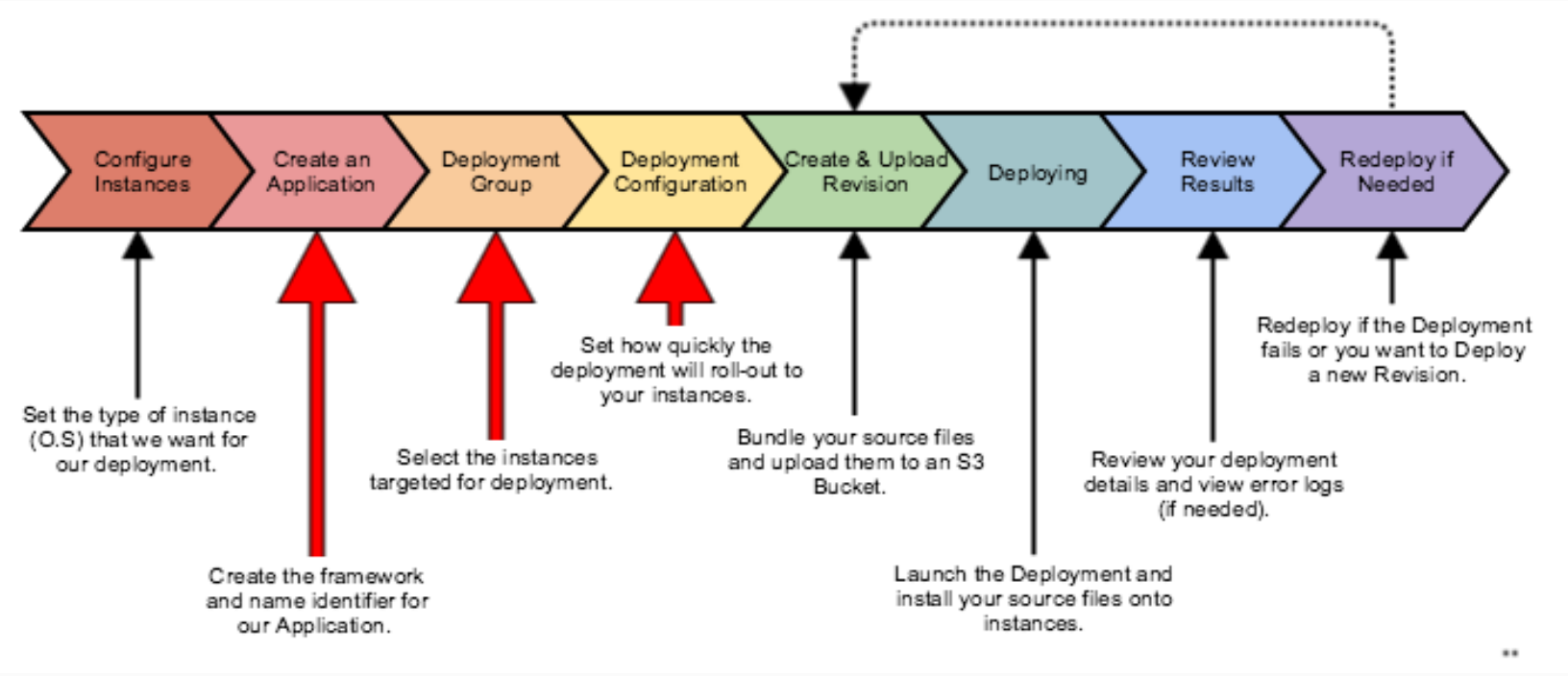
## CodeDeploy: Editing and Deleting an Application

# Thank you for watching!


# Amazon Web Services

## CodeDeploy: Editing, Adding & Deleting Deployment Groups & Configurations

# CodeDeploy Workflow:



# Edit, Add & Delete Deployment Groups & Configurations (in the AWS Console):

- Editing and Adding Groups & Configurations:
  1) *Navigate to CodeDeploy*
  2) *Click on one of your applications*
  3) *Select a deployment group and under actions, select "Edit"*
  4) *Here you can:*
     - *Change the deployment group name*
     - *Add/change/delete instances to deploy to*
     - *Change the deployment configurations*
     - *Add/edit/remove SNS Triggers*
     - *Change the service role*

- Adding an Additional Deployment Group:
  1) Click on an application
  2) Under Deployment Groups, click 'Create Deployment Group'
  3) Fill in the form with new information

- Deleting an Deployment Group:
  1) *Click on an application*
  2) Under Deployment Groups, select a deployment group
  3) Under Actions, click 'Delete'

# Edit, Add & Delete Deployment Groups & Configurations (using the AWS CLI):

- Base CLI Command:
  ***aws deploy***

- Edit a deployment group/configuration elements:
  *required arguments:*
  ***aws deploy update-deployment-group --application-name <NAME> --current-deployment-group-name <NAME>***

  *Optional (elements to change):*
  ***--new-deployment-group-name <NAME>***
  ***--ec2-tag-filters***
  ***--on-premises-instance-tag-filters Key=Name,Value=<VALUE>,Type=KEY_AND_VALUE***
  ***--auto-scaling-groups <NAME>***
  ***--deployment-config-name CodeDeployDefault.<SELECTOPTION>***
  ***--service-role-arn <SERVICE-ROLE_ARN>***

- Delete a deployment group/configuration:
  **aws deploy delete-deployment-group --application-name <NAME> --deployment-group-name <NAME>**

Linux Academy

# Amazon Web Services

## CodeDeploy: Editing, Adding & Deleting Deployment Groups & Configurations

# Thank you for watching!


# Amazon Web Services

## CodeDeploy: Creating & Deleting a Custom Deployment Configuration

# Creating & Deleting Custom Deployment Configurations:

- Default AWS Deployment Configurations:
  - *OneAtATime*
  - *AllAtOnce*
  - *HalfAtATime*

- Creating a Custom Deployment Configuration:
  - *You can create a custom configuration via:*
    - *The AWS CLI*
    - *The AWS APIs*
    - *AWS CloudFormation Template*
  - *CLI Command:*

    ***aws deploy create-deployment-config***

    *Options*

    ***--deployment-config-name <NAME>***

    ***--minimum-healthy-hosts type=<OPTION>,value=<# or %>***
    - *HOST_COUNT*
    - *FLEET_PERCENT*

# Creating & Deleting a Custom Deployment Configuration:

- Viewing a Custom Deployment Configuration:
  - *CLI Commands:*

    *To list all Deployment Configurations*
    ***aws deploy list-deployment-configs***

    *To view detailed information on a specific Configuration*
    ***aws deploy get-deployment-config --deployment-config-name <NAME>***

- Deleting a Custom Deployment Configuration:
  - CLI Command:

    ***aws deploy delete-deployment-config --deployment-config-name <NAME>***


# Amazon Web Services

## CodeDeploy: Creating & Deleting a Custom Deployment Configuration

# Thank you for watching!

Linux Academy

# Amazon Web Services

## CodeDeploy: Creating & Configuring the AppSpec File

# What is the AppSpec File?

- AppSpec (short for Application Specification):
  - *Is YAML-formatted file used to specify the:*
    - *Source and target location of the files we want to deploy*
    - *Permissions given to your files once at their target location*
    - *Lifecycle event hooks available to run specific scripts against*
  - *The AppSpec file MUST be named **"appspec.yml"***

- YAML (YAML Ain't Markup Language):
  - *Is a human friendly data serialization standard for all programming languages*

- Source/Target location (required):
  - *The location of the directory (or specific file) we want to deploy from, and the target directory (or specific file location) we want to deploy to.*

- *Permissions (optional)*
  - *Specifies what (if any) special permissions you want your file or directories to have once deployed in the instance*

***Only applies to Amazon Linux, Ubuntu Server & RHEL***

- *LifeCycle Event Hooks (optional):*
  - *Events in the deployment lifecycle that can trigger specific scripts to run*

# AppSpec File “Header” Section:

- ***version***:
  *This number is completely arbitrary and can be used by you to keep track of versions/revisions.*
- ***os***:
  *This is where you will specify the operating system of the instances you are deploying to. There are only two options, and you must choose only one of them.*
  *1) linux*
  *2) windows*

- Example:
  ***version:*** *1.0*
  ***os:*** *linux*

  ***version:*** *2.0*
  ***os:*** *windows*

# AppSpec File “Files” Section:

- Executes during the deployments “Install” lifecycle event.
- Tells CodeDeploy which files from your application revision should be installed during deployment.

- Base Format:

***files:***
***- source:*** **<***source-file-location***>**
***destination:*** **<***destination-file-location***>**

- Example:

***files:***
***- source:*** */html/wonderwidgets.html*
***destination:*** */mywebsitefiles* *(c:\mywebsitefiles)*
***- source:*** */executables*
***destination:*** */exe* *(c:\exe)*

- Source Options:
  - *If ‘source’ refers to a file, ONLY that file will be installed*
  - *If ‘source’ refers to a directory, ALL directory content will be installed*
  - *If ‘source’ is just a single ‘ / ’, ALL files in the Revision will be installed*

# AppSpec File “Permissions” Section:

- Specifies and assigns any special permissions you would like to assign to a file, files, or directory after installation.
- This section is ***optional*** and only applies to Amazon Linux, Ubuntu Server, and RHEL instances.
- This section MUST be removed for windows servers.

- Base Format:

***permissions:***
  ***- object:*** *<object-specification>*
    ***pattern:*** *<pattern-specification>*
    ***except:*** *<exception-specification>*
    ***owner:*** *<owner-account-name>*
    ***group:*** *<group-name>*
    ***mode:*** *<mode-specification>*
    ***acls:***
      ***-*** *<acls-specification>*
    ***context:***
      ***user:*** *<user-specification>*
      ***type:*** *<type-specification>*
      ***range:*** *<range-specification>*
    ***type:***
      ***-*** *<object-type>*

# AppSpec File "Permissions" Section:

- ***object*** *(required):*
  *The target directory or file(s) you want to set permissions on.*
- ***pattern****:*
  *Specify a pattern to identify certain type of files to apply permissions to.*
- ***except****:*
  *Specify exceptions to the above pattern*
- ***owner****:*
  *Set the owner for the object (source settings if blank)*
- ***group****:*
  *Set the group for the object (source settings if blank)*
- ***mode****:*
  *Set the permissions value (think chmod command)*
- ***acls****:*
  *Access Control List entries applied to the object.*
- ***context****:*
  *For Security-Enhanced Linux (SELinux)-enabled instances.*
  - *user: The SELinux user*
  - *type: The SELinux type name*
  - *range: The SELinux range specifier*
- ***type****:*
  *Specify if the object is a file or a directory*
  - *file: Permissions will be applied only to object's files*
  - *directory: Permissions will be applied recursively to all directories, and files in the object.*

# AppSpec File “Permissions” Section:

- Example:

```
permissions:
  - object: /wonderwidgets/html
    pattern: “*.html”
    except: “index.html”
    mode: 400
    type:
      - file
```

- *acls & context:*

```
    acls:
      - u:matt:rw
      - u:kelly:rwx
    context:
      user: unconfined_u
      type: httpd_sys_content_t
      range: s0
```

# AppSpec File "Hook" Section:

- Allows you to link deployment lifecycle event hooks to scripts.

- Deployment Lifecycle Event Hooks:
  - ***ApplicationStop*** *(Before app revision is downloaded, can't be used to on the first revision upload)*
  - *DownloadBundle* *(CodeDeploy Agent copies files to a temp loc.)*
  - ***BeforeInstall*** *(Time to run pre-install tasks, i.e. decrypting files or backing up the current version)*
  - *Install* *(CodeDeploy Agent installs files from temp loc. to your destination folder)*
  - ***AfterInstall*** *(Time to configure your application or change permissions on files)*
  - ***ApplicationStart*** *(Restart services that were stopped during ApplicationStop)*
  - ***ValidateService*** *(Verify deployment was competed successfully)*

***highlighted hooks are ones you can run scripts against***

# AppSpec File “Hook” Section:

- Base Format:
  ***hooks:***
  *<deployment-lifecycle-event-name>*
  ***- location:*** *<script-location>*
  ***timeout:*** *<timeout-in-seconds>*
  ***runas:*** *<user-name>*

- ***location*** *(required):*
  *The location of the script file that you want to run.*
- ***timeout****:*
  *The amount of time you want to “allow” the script to run before it is considered to have failed (if not fully competed). Default timeout is set to 3600 second (1 hour).*
- ***runas****:*
  *The user to “assume” when running the script*

# AppSpec File “Hook” Section:

- The scripts that run during the deployment lifecycle can also access the following environment variables:

- ***APPLICTION_NAME****:*
  *The current CodeDeploy Application name.*
- ***DEPLOYMENT_ID****:*
  *An ID number that CodeDeploy assigned to the current deployment.*
- ***DEPLOYMENT_GROUP_NAME****:*
  *The name of the current CodeDeploy Deployment Group.*
- ***DEPLOYMENT_GROUP_ID****:*
  *An ID number that CodeDeploy assigned to the current Deployment Group.*
- ***LIFECYCLE_EVENT****:*
  *The name of the current deployment lifecycle event.*
  ****These environment variables are local to each lifecycle event.***

# AppSpec File “Hook” Section:

- Example:

***hooks:***

*AfterInstall:*

***- location:*** *Scripts/RunResourceTests.sh*

***timeout:*** *180*

# Formatting your AppSpec File:

- Specific spacing is required for the AppSpec file (YAML):

*version:[1]version-number*
*os:[1]operating-system-name*
*files:*
*[2]-[1]source:[1]source-files-location*
*[4]destination:[1]destination-files-location*
*permissions:*
*[2]-[1]object:[1]object-specification*
*[4]pattern:[1]pattern-specification*
*[4]except:[1]exception-specification*
*[4]owner:[1]owner-account-name*
*[4]group:[1]group-name*
*[4]mode:[1]mode-specification*
*[4]acls:*
*[6]-[1]acls-specification*
*[4]context:*
*[6]user:[1]user-specification*
*[6]type:[1]type-specification*
*[6]range:[1]range-specification*
*[4]type:*
*[6]-[1]object-type*
*hooks:*
*[2]deployment-lifecycle-event-name:*
*[4]-[1]location:[1]script-location*
*[6]timeout:[1]timeout-in-seconds*
*[6]runas:[1]user-name*

# Validating your AppSpec File:

- You use a YAML validator to validate the AppSpec File.
- The validator will check to make sure that you have the proper amount of spaces per line and correctly formatted sections.

Linux Academy

# Amazon Web Services

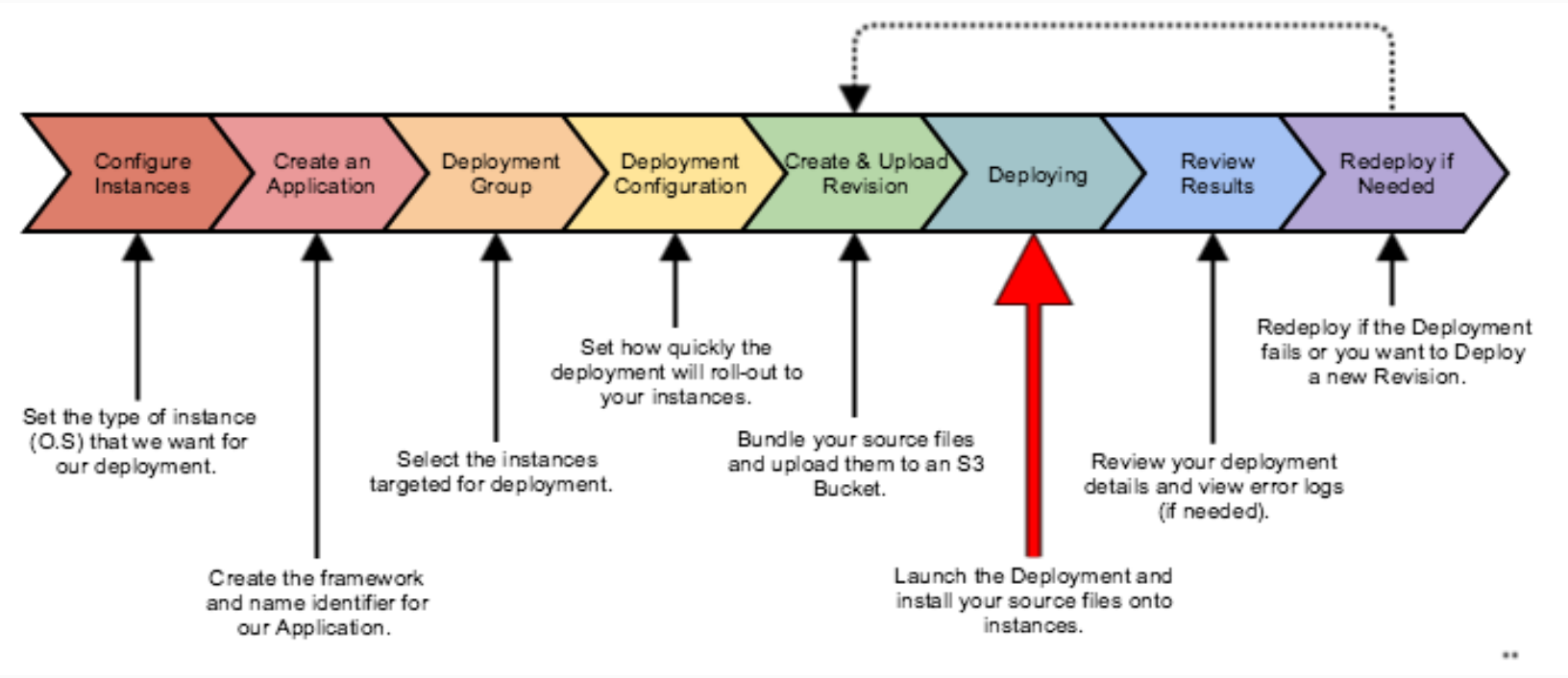
## CodeDeploy: Creating & Configuring the AppSpec File

# Thank you for watching!


# Amazon Web Services

## CodeDeploy: Deploying a Revision

# CodeDeploy Workflow:



# Deploying a Revision:

- You can deploy a revision to an instance via:
  - *AWS Console.*
  - *AWS CLI*
  - *AWS API*

***In this lesson, we will cover how to deploy using both the Console and CLI.***

# Deploying a Revision via the AWS Console:

1) Navigate to CodeDeploy

2) From the dropdown at the top select “Deployments”

3) Click “Create New Deployment”

4) Fill out the form
   - *Application Name*
   - *Deployment Group Name*
   - *Revision Type (Where you revision is stored: S3/Github)*
   - *Revision Location*
   - *Deployment Description*
   - *Deployment Config*

5) Click “Deploy Now”

6) Monitor deployment “status”

7) Done!

# Deploying a Revision via the AWS CLI:

1) Copy and paste the response given when we pushed the revision to our S3 bucket

2) Fill in the required areas inside the <> brakets:
   - *Deployment Group Name*
   - *Deployment Configuration Name:*
     - ***CodeDeployDefault.OneAtATime***
     - ***CodeDeployDefault.AllAtOnce***
     - ***CodeDeployDefault.HalfAtATime***
     - ***Custom (if you created one)***
   - *Description*

3) Execute the command

4) Check the “status” of the deployment:
   - Run the command:

     **aws deploy get-deployment --deployment-id <ID>**

5) Done!

Linux Academy

# Amazon Web Services

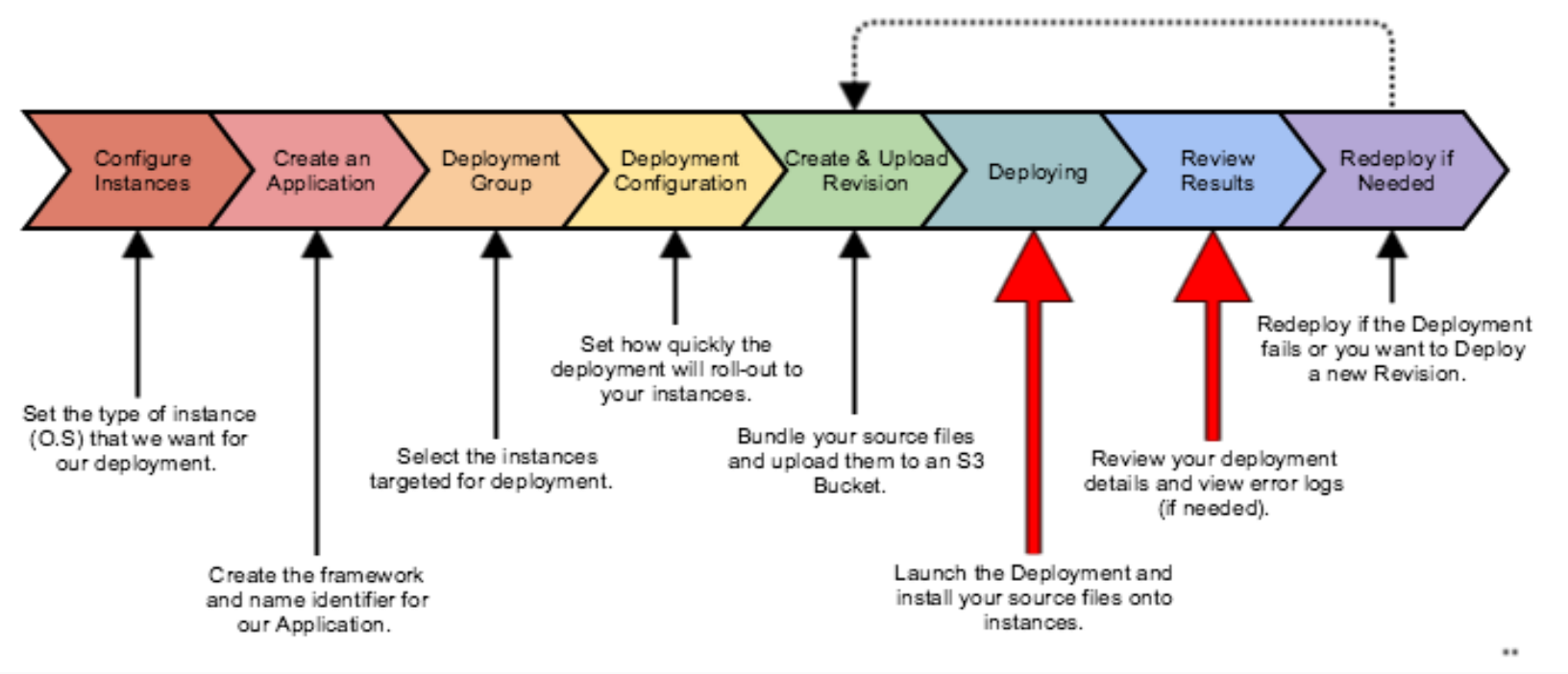
## CodeDeploy: Deploying a Revision

# Thank you for watching!


# Amazon Web Services

## CodeDeploy: Monitoring a Deployment with SNS Triggers

# CodeDeploy Workflow:



# CodeDeploy SNS Triggers:

- SNS Triggers must be setup pre-deployment
- SNS Triggers are great for automating the monitoring process. Allowing you to get instant notifications if deployments or instances fail or succeed.
- SNS Triggers can be setup via:
  - *AWS Console*
  - *AWS CLI*
- SNS Triggers:
  - *Deployment Status:*
    - ***Deployment Starts***
    - ***Deployment succeeds***
    - ***Deployment fails***
    - ***Deployment Stops***
  - *Instance Status:*
    - ***Instance Starts***
    - ***Instance Succeeds***
    - ***Instance Fails***

# Add SNS Triggers via the AWS Console:

1) Navigate to CodeDeploy

2) Click on the Application you want to add a trigger to

3) Click on the “arrow” on the Deployment Group that you want to add the trigger to (this will display the Deployment Group setting)

4) Click on “Create Triggers”

5) Fill our the form, setting or selecting:
   - *Trigger Name*
   - *The event(s) you want to invoke the trigger*
   - *The SNS topic you want to execute*

   ***You must already have a SNS topic setup***

6) Click “Create Trigger”

***You can easily edit or delete triggers under the “triggers” section of the deployment group settings by clicking on the “pencil” icon to edit, or the “x” to delete.***

# Creating an SNS Triggers via the AWS CLI:

- When creating a new Deployment Group:

1) Run the command:
***aws deploy create-deployment-group --generate-cli-skeleton***

2) Fill out JSON file will all the info for the Deployment Group

3) Upload the JSON file to create the new Deployment Group, which will include the SNS Trigger you added, by using the command:
***aws deploy create-deployment-group --cli-input-json file://<FILENAME>.json***

# Creating an SNS Triggers via the AWS CLI:

- Adding a Trigger to an existing Deployment Group:

1) Run the command:
***aws deploy get-deployment-group --application-name <APP-NAME> --deployment-group-name <DEPLOYMENT-GROUP-NAME>***

2) Copy the entire JSON text block

3) Create a .json file and open it with a text editor

4) Copy the JSON text block into the text editor

5) Remove the following items:
   - *"deploymentGroupInfo": {*
   - *"deploymentGroupId": "XXXX",*
   - *"deploymentGroupName": "XXXX",*
   - *The entire "targetRevision" section (if your file has it)*
   - *The remaining "}" from when you deleted "deploymentGroupInfo" above*

# Creating an SNS Triggers via the AWS CLI:

- Adding a Trigger to an existing Deployment Group:

6) Add the following JSON text:

```
"triggerConfigurations": [
  {
      "triggerEvents": [

      ],
      "triggerTargetArn": "",
      "triggerName": ""
      }
],
```

7) Add one or more Events to “triggerEvents”
   - *“DeploymentStart”*
   - *“DeploymentSuccess”*
   - *“DeploymentFailure”*
   - *“DeploymentStop”*
   - *“InstanceStart”*
   - *“InstanceSuccess”*
   - *”InstanceFailure”*

# Creating an SNS Triggers via the AWS CLI:

- Adding a Trigger to an existing Deployment Group:

8) Add the Topic ARN for the SNS Topic you want to invoke to “triggerTargetArn”:

9) Add a name to “triggerName”:

10) Save and exit

11) Upload the .json file using the command:
***aws deploy update-deployment-group --current-deployment-group-name <DEPLOYMENT-GROUP-NAME> --cli-input-json file://<FILENAME>.json***

# Creating an SNS Triggers via the AWS CLI:

- Viewing, Modifying & Deleting a Trigger in an existing Deployment Group:

1) To view the triggers in a Deployment Group, run:
***aws deploy get-deployment-group --application-name <APP-NAME> --deployment-group-name <DEPLOYMENT-GROUP- NAME>***

2) To modify or delete triggers:
   - *Open the .json file:*
     - *Add events*
     - *Remove events*
     - *Delete the entire contents of the "trigger section"*

3) Save and exit

4) Re-upload the .json file

Linux Academy

# Amazon Web Services

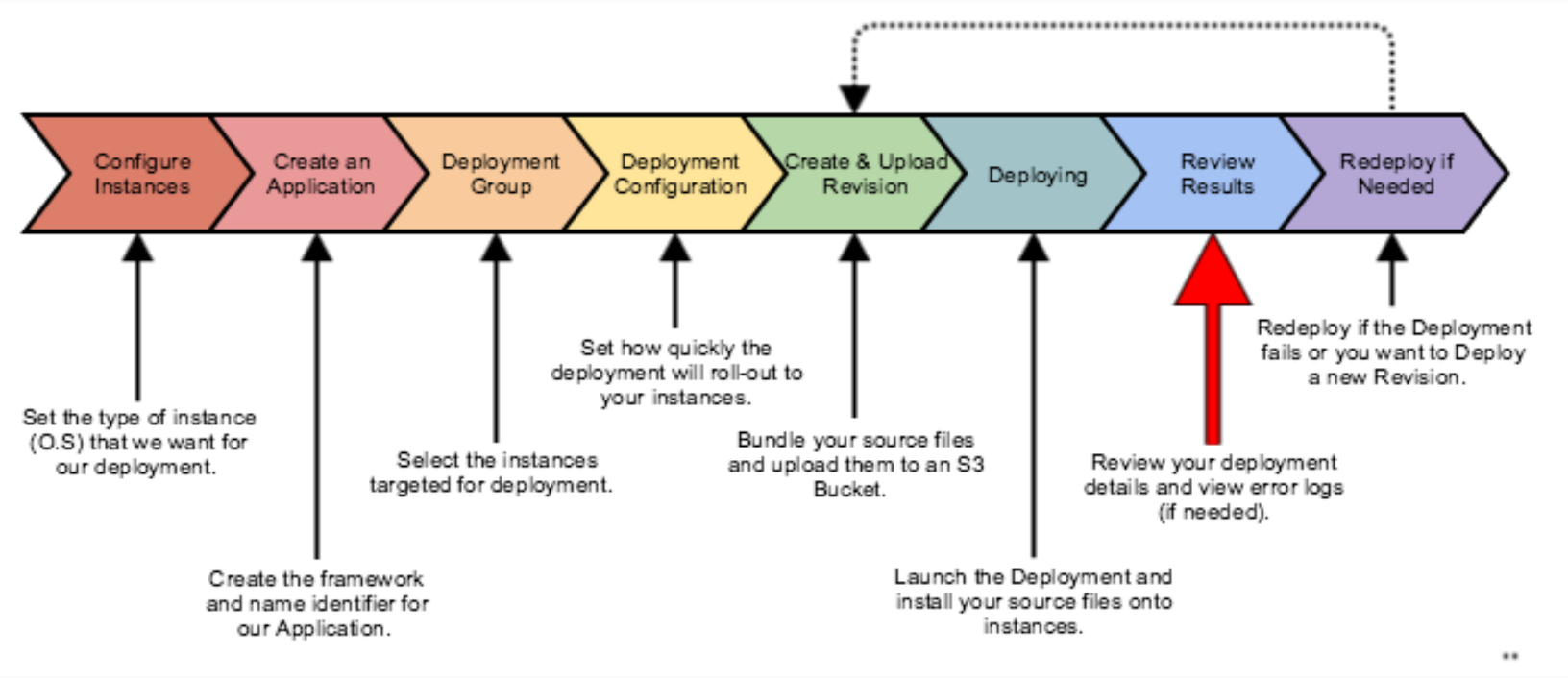
## CodeDeploy: Monitoring a Deployment with SNS Triggers

# Thank you for watching!


# Amazon Web Services

## CodeDeploy: Viewing Deployment Details and Error Logs

# CodeDeploy Workflow:



# Viewing Deployment Details via the Console:

- What can we view?
  - *Deployment details*
  - *Instance details*
  - *Application details*
  - *Deployment Group details*
  - *Application Revision details*
  - *Deployment Configuration details*

- After you make your first deployment, or multiple deployment, I suggest you take some time to click around CodeDeploy in the AWS Console to get familiar with all the various views and information that is available.

# Viewing Deployment Details via the CLI:

- Deployment details:
  - *List Deployment ID numbers:*
    ***aws deploy list-deployments***
    ***--application-name <APP-NAME>***
    ***--deployment-group-name <DEPLOY-GROUP-NAME>***

    *Optional:*
    ***--include-only-statuses <Failed or Succeeded>***
    ***--create-time-range***
    ***start=<2014-08-19T00:00:00>,end=<2014-08-20T00:00:00>***

  - List detailed deployment info:
    **aws deploy get-deployment --deployment-id <ID #>**

- Instance details:
  - *List instances*

***aws deploy list-deployment-instances --deployment-id <DEPLOYMENT ID #>***

  - *List detailed information about an instance*

***aws deploy get-deployment-instance --deployment-id <DEPLOYMENT ID> --instance-id <INSTANCE ID>***

# Viewing Deployment Details via the CLI:

- Application details:
  - *List Applications:*
    ***aws deploy list-applications***
  - List detailed Application info:
    ***aws deploy get-application --application-name <APP-NAME>***

- *Deployment Group details*:
  - *List Deployment Groups:*
    ***aws deploy list-deployment-groups --application-name <APP-NAME>***
  - List detailed Deployment Group info:
    ***aws deploy get-deployment-group --application-name <APP-NAME> --deployment-group-name <DEPLOYMENT-GROUP-NAME>***

# Viewing Deployment Details via the CLI:

- Application Revision details:
  - *List Application Revisions:*
    ***aws deploy list-application-revisions --application-name <APP-NAME>***
  - List detailed Application Revision info:
    ***aws deploy get-application-revision --application-name <APP_NAME> --s3-location bucket=<S3-BUCKET>, bundleType=<TYPE>,eTag=<TAG-VALUE>,key=<FILE-NAME>***

- *Deployment Configuration details*:
  - *List Deployment Configurations:*
    ***aws deploy list-deployment-configs***
  - List detailed *Deployment Configuration* info:
    ***aws deploy get-deployment-config --deployment-config-name <CONFIG-NAME>***

Linux Academy

# Amazon Web Services

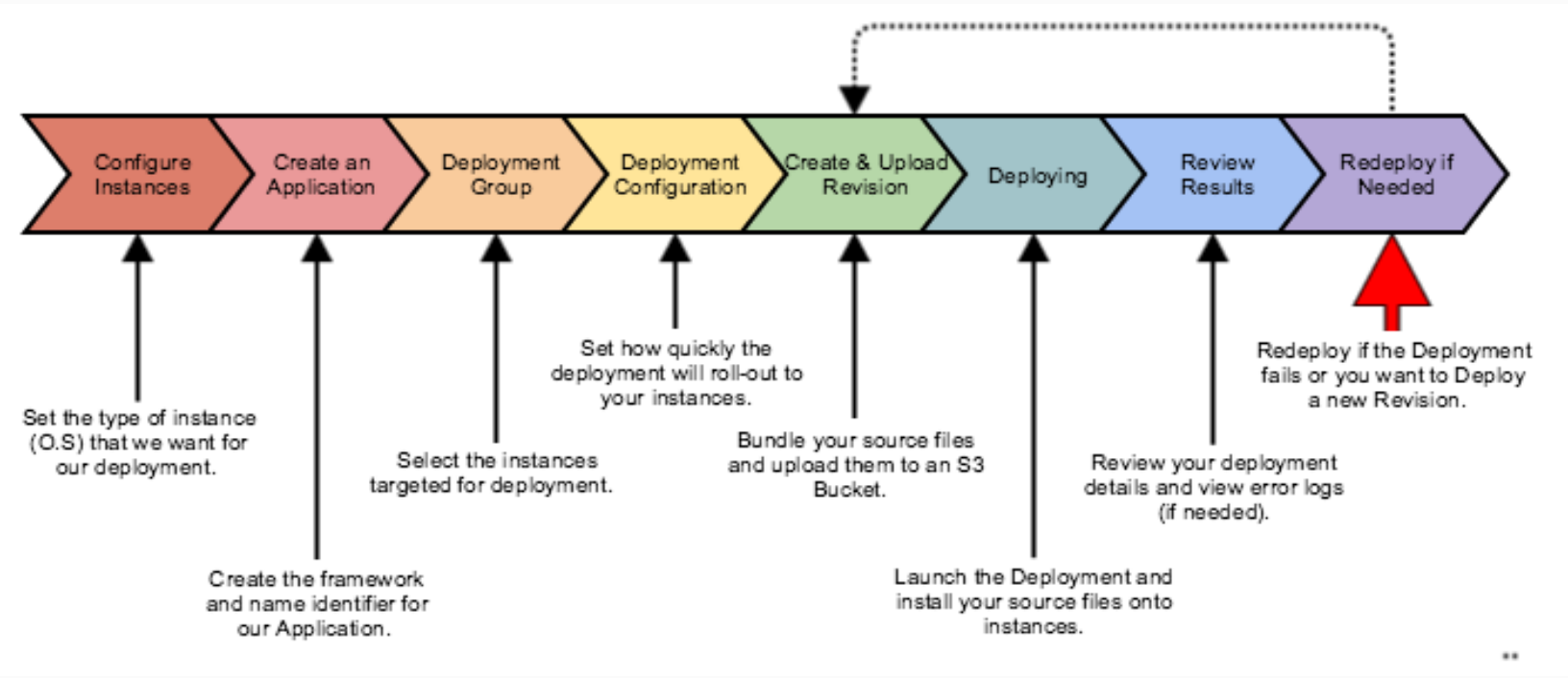
## CodeDeploy: Viewing Deployment Details and Error Logs

# Thank you for watching!


# Amazon Web Services

## CodeDeploy: Stopping, Rolling-Back and Redeploying a Revision

# CodeDeploy Workflow:



# Stopping a Deployment:

- AWS Console:
  1) *While a deployment is deploying, be on the "Deployments" page in CodeDeploy.*
  2) *Click the "Stop" button located in the **Actions** column (for the deployment you want to stop)*

- AWS CLI:
  1) *While a deployment is deploying, run the command:* ***stop-deployment --deployment-id <DEPLOYMENT-ID-#>***

- State of your Instances?
  - *Partial or full file installation*
  - *Partial or all scripts run*
  - *Nothing run or installed*

***This can really depend on what Deployment Configuration you are using, and how many instances you are deploying to**.*

# Rolling-Back & Redeploying:

- Understanding Roll-Back and Redeployments:
  1) *CodeDeploy treats a "Redeployment" as just a deployment of an already deployed Revision.*
  2) *CodeDeploy does not have an automatic "Roll-back" feature.*

- "Cleanup" file:
  - *The cleanup files is where CodeDeploy stores information about the last installed files, so that in the case of a redeployment, CodeDeploy can try to delete those files.*

  - *File location (Linux):*
    ***/opt/codedeploy-agent/deployment-root/deployment instructions/<DEPLOYMENT-GROUP-ID>-cleanup***

  - *File location (Windows):*
    **C:\ProgramData\Amazon\CodeDeploy\deployment instructions\<DEPLOYMENT-GROUP-ID>-cleanup**

Linux Academy

# Amazon Web Services

## CodeDeploy: Stopping, Rolling-Back and Redeploying a Revision

# Thank you for watching!


# Amazon Web Services

## CodeDeploy: Automating Deployments From an S3 Bucket using Lambda

# Process overview:

1) Create a Lambda “execution role”:
   - *Give Lambda permission to access S3 & CodeDeploy*

2) Configure a Lambda Function:
   - Prepare the code for the Lambda Function

3) Register a Lambda Function:
   - Setup the Lambda Function in AWS

4) Bundle the source files into a .zip file:

5) Upload the .zip file the an S3 bucket:

# Creating the Lambda "Execution Role":

1) Create a custom IAM policy:
   - *For your convenience, the policy can be found in the Study Guide section of this course*
   - *Enter the appropriate S3 Bucket Name, Region & Account ID # into the template*

2) Create a new Role:
   - *For Role Type, select "AWS Lambda"*
   - *Attach the custom policy you just created to this role*

# Creating the Lambda Function:

1) Get the Lambda Function code:
   - *For your convenience, the script containing the code can be found in the Study Guide section of this course*

2) Create a Lambda Function:
   - *Navigate to Lambda in the AWS Console*
   - *Click "Create a Lambda Function"*
   - *Click "Skip"*
   - *Give the function a name, and copy the script into the "code" section of the form*
   - *For "Role" select the Lambda Role you just created*
   - *Leave everything else as default and click "Next"*
   - *Click "Create Function"*

*3)* Add an Event Source to the Lambda Function:
   - *Set Source Type = S3*
   - *Set Bucket = your target S3 Bucket*
   - *Set Event Type = Object Created (All)*
   - *Make sure that "Enabled event source" is checked*
   - *"Submit" to add the event source*

# Preparing and Uploading the Source Files:

1) Creating a .zip file of the source files:
   - *Make sure you appspec.yml file is moved outside your root application folder (should be one directory up)*
   - *Install zip if you don't already have it installed*
   - *Run the following command to properly create the zip file:*

***zip -r <NAME-OF-ZIP-FILE>.zip appspec.yml <APP-DIRECTORY> -x "*/\.*"***

2) Upload the zip file to S3:
   - *Run the S3 PUT command:*

***aws s3api put-object --bucket <S3-BUCKET-NAME> --key <ZIP-FILE-NAME>.zip --body <ZIP-FILE-NAME>.zip --metadata application-name=<APPLICATION-NAME>, deploymentgroup-name=<DEPLOYMENT-GROUP-NAME>***

Linux Academy

# Amazon Web Services

## CodeDeploy: Automating Deployments From an S3 Bucket using Lambda

# Thank you for watching!