# A tutorial on AWS CodeDeploy

Xin Yin

10/14/2015
Cloud Computing Working Group

# CodeDeploy - Examples

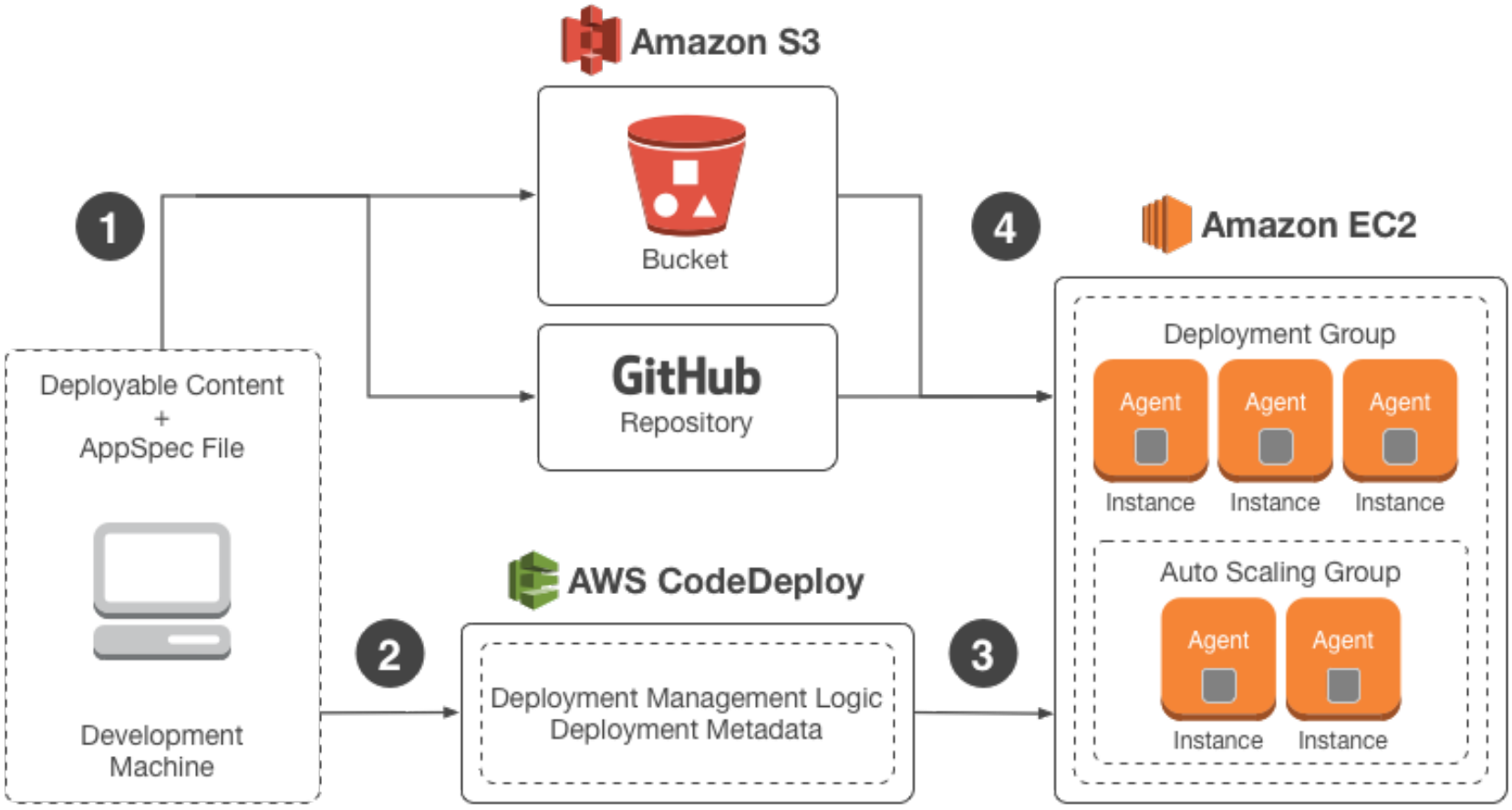
1. Alice runs a few Shiny servers on multiple EC2 instances. Now that Alice wants to deploy a new version of her Shiny application to all servers.
2. Bob wants to run some bioinformatics tools on multiple EC2 instances to crunch his sequencing data in parallel. To do so, he needs to install some packages and software.
3. Charlie uses multiple EC2 instances to run his simulations. He made some changes to his R code and wants to upload and rerun the analysis with the updated R program.
4. Daniel sporadically uses spot instances for computing, and he wants to automate the process of installing R, R packages and upload his R program to newly launched instances.

# CodeDeploy - Examples

1. Alice runs a few Shiny servers on multiple EC2 instances. Now that Alice wants to deploy a new version of her Shiny application to all servers.
2. Bob wants to run some bioinformatics tools on multiple EC2 instances to crunch his sequencing data in parallel. To do so, he needs to install some packages and software.
3. Charlie uses multiple EC2 instances to run his simulations. He made some changes to his R code and wants to upload and rerun the analysis with the updated R program.
4. Daniel sporadically uses spot instances for computing, and he wants to automate the process of installing R, R packages and upload his R program to newly launched instances.

**NOTE**: the solutions are not **unique** (AMIs, Docker, `rsync`, `scp`, Git-Hooks, `user-data`).

# CodeDeploy in a nutshell




http://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html

# CodeDeploy - Terminology

- Application:
  *"An application in AWS CodeDeploy is simply a unique identifier used by AWS CodeDeploy to deploy the correct revision to the correct set of instances with the correct deployment configuration."*
- Deployment Group:
  A deployment group specify a set of instances to which a **revision** of your application/code is deployed, according to deployment configurations.
- Revision:
  A **snapshot** of your code-repository/application at a certain version.

# CodeDeploy - A quick tutorial

## 0. Steps

1. Create IAM **Roles** and **Policies**
2. Launch and configure your EC2 instances
3. Create an **Application** and **Deployment Group**
4. Prepare a **Revision**
5. Deploy!

You can also check out AWS's tutorial:

`http://docs.aws.amazon.com/codedeploy/latest/userguide/getting-started-walkthrough.html`

# CodeDeploy - A quick tutorial

## 1. Configure IAM Roles and Policies

- Our goal is to create **two roles**, each attached with certain **policies** (permitting some actions to be performed).
- Roles can be *assumed* by certain services (*e.g.* EC2, CodeDeploy). By assuming the role, the service gains the attached **policies**.
  - *CodeDeploy* assumes some role to interact with the instances.
  - Your *EC2 instances* assumes some other role to interact with *AWS S3*.

# CodeDeploy - A quick tutorial

## 1. Configure IAM Roles and Policies

- Our goal is to create **two roles**, each attached with certain **policies** (permitting some actions to be performed).
- Roles can be *assumed* by certain services (*e.g.* EC2, CodeDeploy). By assuming the role, the service gains the attached **policies**.
  - *CodeDeploy* assumes some role to interact with the instances.
  - Your *EC2 instances* assumes some other role to interact with *AWS S3*.
- Live Demo

Reference: `http://docs.aws.amazon.com/codedeploy/latest/userguide/how-to-create-service-role.html`

# CodeDeploy - A quick tutorial

## 2. Launch EC2 Instances and Tag Them

- Your code will be deployed to EC2 instances (on-demand, spot, reserved) launched in specific region(s).
- To let CodeDeploy discover your EC2 instances, **tag them**.
  - You can attach up to 10 tags to an instance.
  - Tags are simply *key-value* pairs, *e.g.* `deploy-r=true`, `app=shinydemo`.
  - If you launch spot instances using the AWS Console, you **must** tag instances after your requests are fulfilled.
- Utilize the *user-data* of your EC2 instance to install CodeDeploy agent on the newly launched instances automatically.

# CodeDeploy - A quick tutorial

## 3. Create an Application and Deployment Group

- Don't confuse the "*application*" here with your actual "program" or "code-repository". Here, "application" merely refers to a unique identifier in the CodeDeploy service.
- Deployment group uses **tags** to identify a group of instances to which your program is deployed.

# CodeDeploy - A quick tutorial

## 3. Create an Application and Deployment Group

- Don't confuse the "*application*" here with your actual "program" or "code-repository". Here, "application" merely refers to a unique identifier in the CodeDeploy service.
- Deployment group uses **tags** to identify a group of instances to which your program is deployed.
- Live Demo

Reference: `http://docs.aws.amazon.com/codedeploy/latest/userguide/how-to-create-application.html`

# CodeDeploy - A quick tutorial

## 4. Prepare a Revision

- A **revision** refers to the collection of:
  1. Your program (source code, executables, supporting libraries, configurations)
  2. Deployment scripts/hooks
  3. `appspec.yml`
- To prepare a revision, arrange files in a directory like this:

```
/home/alice/

  |--shinydemo
  |     |--server.R
  |     |--ui.R
  |     |--README.md
  |     |--static
  |     |     |--foo.js

  |--scripts
  |     |--before_install.sh
  |     |--after_install.sh

  |--appspec.yml
```

# CodeDeploy - A quick tutorial

## 4. Prepare a Revision (appspec.yml)

- `os`: `linux` or `windows`
- `files`: a *mapping* to tell CodeDeploy agent, **which files** to copy, and **to where**
- `permissions`: **who** should own the files? what are the permissions?
- `hooks`: what scripts should the CodeDeploy agent execute during the deployment

```
version: 0.0
os: linux

files:
 - source: shinydemo
   destination: /home/ubuntu

permissions:
  - object: shinydemo
    pattern: "**"
    owner: ubuntu
    mode: 644

hooks:
  BeforeInstall:
    - location: scripts/install_R.sh
      timeout: 180
  ApplicationStart:
    - location: scripts/run_shiny_server.sh
      timeout: 60
      runas: ubuntu
```

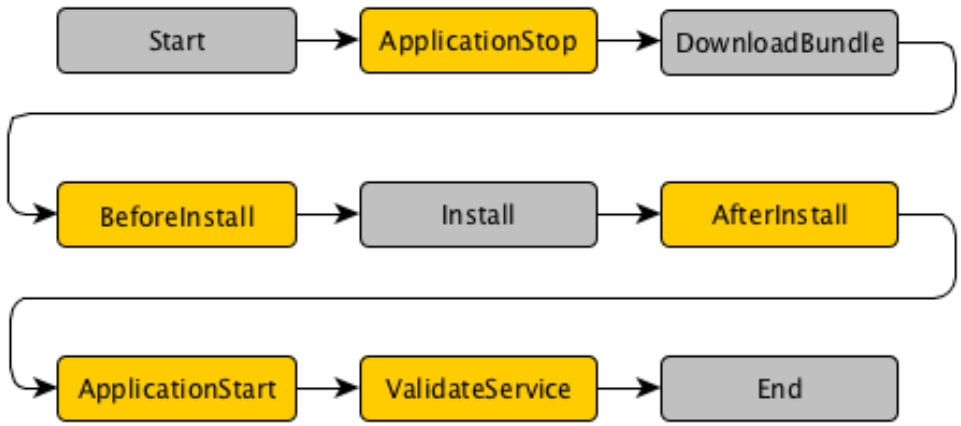
YAML Syntax: `http://learn.getgrav.org/advanced/yaml`

appspec.yml reference: `http://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html`

# CodeDeploy - A quick tutorial

## 4. Prepare a Revision (continued)

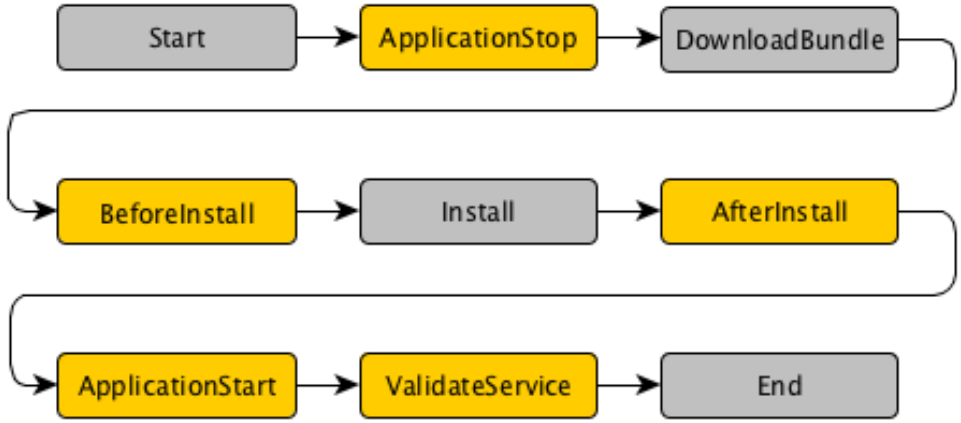
- **Hooks** section defines scripts that will be executed during various lifecycle of a deployment process.
- You can hook up scripts with only the yellow events.
- Using scripts can customize your deployment with a lot of flexibility.



# CodeDeploy - A quick tutorial

## 4. Prepare a Revision (continued)

- **Hooks** section defines scripts that will be executed during various lifecycle of a deployment process.
- You can hook up scripts with only the yellow events.
- Using scripts can customize your deployment with a lot of flexibility.




Live Demo

`http://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html#app-spec-ref-hooks`

# CodeDeploy - A quick tutorial

## 5. Deploy and Monitor the Deploying Process

- To deploy the prepared revision, we can put the directory into a `.zip` or `.tar.gz` archive.
- Upload the archive to AWS S3 or GitHub.
- Deploy it!

# CodeDeploy - A quick tutorial

## 5. Deploy and Monitor the Deploying Process

- To deploy the prepared revision, we can put the directory into a `.zip` or `.tar.gz` archive.
- Upload the archive to AWS S3 or GitHub.
- Deploy it!
- Live Demo

To see how to deploy your code on GitHub, please refer to

`http://docs.aws.amazon.com/codedeploy/latest/userguide/how-to-deploy-revision.html`

# Deploy the same program, but run with different parameters?

- Use instance meta-data.
- Use instance tags and `ec2-describe-tags`.