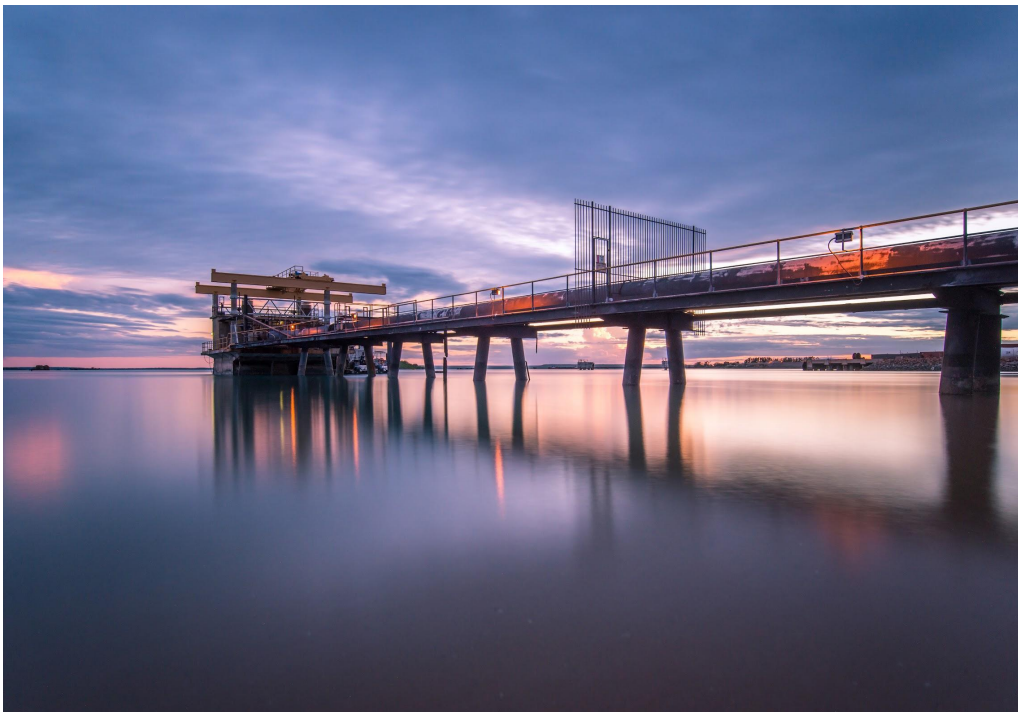


# The risk of CI/CD pipeline poisoning via CodeBuild

On the intricate  
challenges of setting up a  
secure CI/CD pipeline



# Hello! I am Asier Rivera Fernandez

Technical expert - Cybersecurity researcher at PwC Belgium

 <https://www.linkedin.com/in/asier-rivera-fernandez/>



I grew up in a charming and intriguing area in the north of Spain, called Euskal Herria (*Basque Country*)



I studied about computer science and security in Spain, Sweden and Belgium



I enjoy brainstorming and bringing crazy ideas to the table until my brain hurts



Do we  
understand  
cloud services  
well enough?

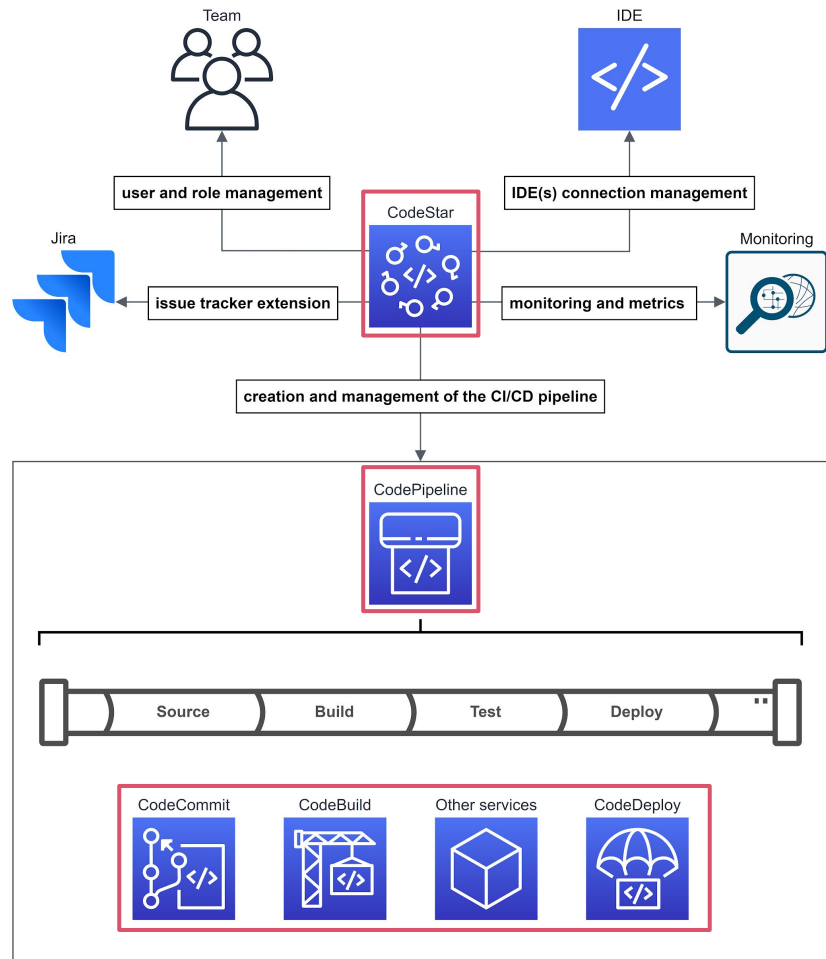
**Deep dive into CI/CD pipelines in the cloud**

**Risk analysis**

**Threat modelling**

**Tailor CI/CD pipeline security in the cloud**

# CI/CD pipelines in AWS, the Code services



source package

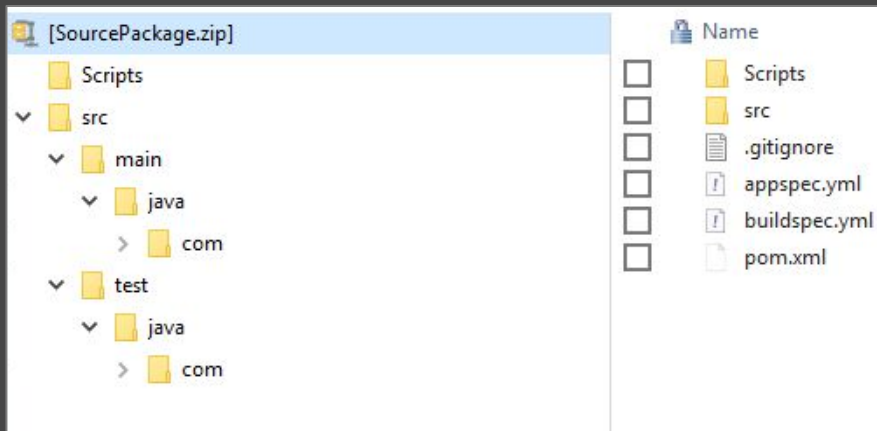


# Source package

Created by CodeCommit / Used by CodeBuild

Resources required to compile and configure the application:

- Source code
- Media files
- Configuration files
- Compilation settings



artifact package

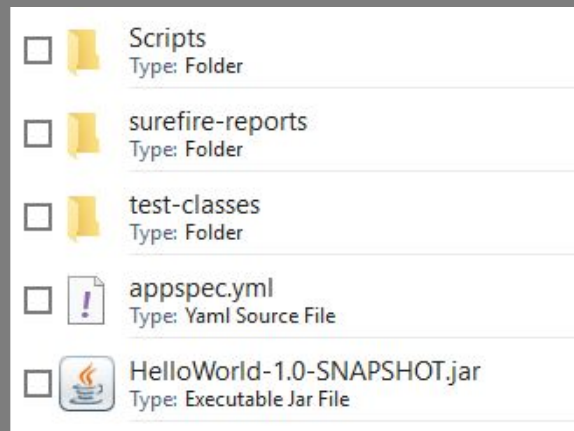


# Artifact package

Created by CodeBuild / Used by CodeDeploy

Everything required to install the new version of the application:

- The Scripts to be executed during installation (optional)
- Resources and others
- AppSpec file (required by CodeDeploy)
- Application





# BuildSpec file

Used by CodeBuild

CodeBuild specific file:

- Contains the commands to be run in the container
- Set as project configuration (optional)
- It can be provided within source code package

```
version: 0.2

env:
  variables:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"
  parameter-store:
    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword

phases:
  install:
    commands:
      - echo Entered the install phase...
      - apt-get update -y
      - apt-get install -y maven
    finally:
      - echo This always runs even if the update or install command fails
pre_build:
```



# AppSpec file

Used by CodeDeploy

CodeDeploy specific file:

- Contains commands to run in the server
- Provided within the Artifact package content
- Sample below is specific to EC2/On-Premises deployment

```
version: 0.0
os: linux
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
hooks:
  BeforeInstall:
    - location: Scripts/UnzipResourceBundle.sh
    - location: Scripts/UnzipDataBundle.sh
  AfterInstall:
```

# Let's simulate a company for a moment

## Roles:

- **Asier: new developer** at the company, I need access rights.
- **Audience: the administrator** and you have been requested to provide the developer with the correct access rights.

## AWS infrastructure:

- Complete CI/CD pipelines in AWS (CodePipeline, CodeStar)
- Deployment in AWS services (EC2, Lambda)

AWS Managed policies	AWS CodeStar Team roles
AWS <u>CodeBuild</u> <b>Admin</b> Access	<b>Owner</b>
AWS <u>CodeBuild</u> <b>Developer</b> Access	<b>Contributor</b>
AWS <u>CodeBuild</u> <b>ReadOnly</b> Access	<b>Viewer</b>

Would you  
allow me to ...

aws

Search in this guide

English ▼

AWS > Documentation > **AWS CodeBuild** > API Reference

Feedback ⓘ Preferences ⚙️

## ListBuilds

[PDF](#)

Gets a list of build IDs, with each build ID representing a single build.

AWS Managed policy for CodeBuild Developer

CodeStar Team's Contributor Role



Would you  
allow me to ...

The screenshot shows the AWS documentation page for the `DeleteProject` API action. The breadcrumb navigation indicates the path: `AWS > Documentation > AWS CodeBuild > API Reference`. The `AWS CodeBuild` link is highlighted with an orange box. The page title is `DeleteProject`, and there is a link to a PDF version. The description states: "Deletes a build project. When you delete a project, its builds are not deleted." Below the documentation, there are two red boxes with white text: "AWS Managed policy for CodeBuild Developer" and "CodeStar Team's Contributor Role".

Would you  
allow me to ...

The screenshot shows the AWS documentation page for the `StartBuild` API. The breadcrumb navigation path is `AWS > Documentation > AWS CodeBuild > API Reference`, with `AWS CodeBuild` highlighted by an orange box. The page title is `StartBuild`, with a `PDF` link below it. The description states: `Starts running a build.` Below the screenshot, two green boxes are displayed: `AWS Managed policy for CodeBuild Developer` and `CodeStar Team's Contributor Role`.

# StartBuild parameters

The screenshot shows the AWS CodeBuild API Reference page for the `StartBuild` operation. The left sidebar lists various API actions, with `StartBuild` highlighted. The main content area displays two parameters: `artifactsOverride` and `buildspecOverride`. Each parameter has a description, type, and required status. Two orange callout boxes are overlaid on the page to provide additional context.

**artifactsOverride**  
Build output artifact settings that override, for this build only, the latest ones already defined in the build project.  
Type: `ProjectArtifacts` object  
Required: No

**buildspecOverride**  
A buildspec file declaration that overrides, for this build only, the latest one already defined in the build project.  
If this value is set, it can be either an alternate buildspec file relative to the environment variable, or the path to an S3 bucket. The bucket must be in the same AWS Region as the build project. Specify the buildspec file using its ARN (for example, `arn:aws:s3:::my-codebuild-sample2/buildspec.yml`). If this value is not provided or is set to an empty string, the source code must contain a buildspec file in its root directory. For more information, see [Buildspec File Name and Storage Location](#).  
Type: String  
Required: No

**Define the location for the output artifact**

**Define the commands to run in the container**

# An AWS developer can abuse StartBuild to ...

1

Exfiltrate data

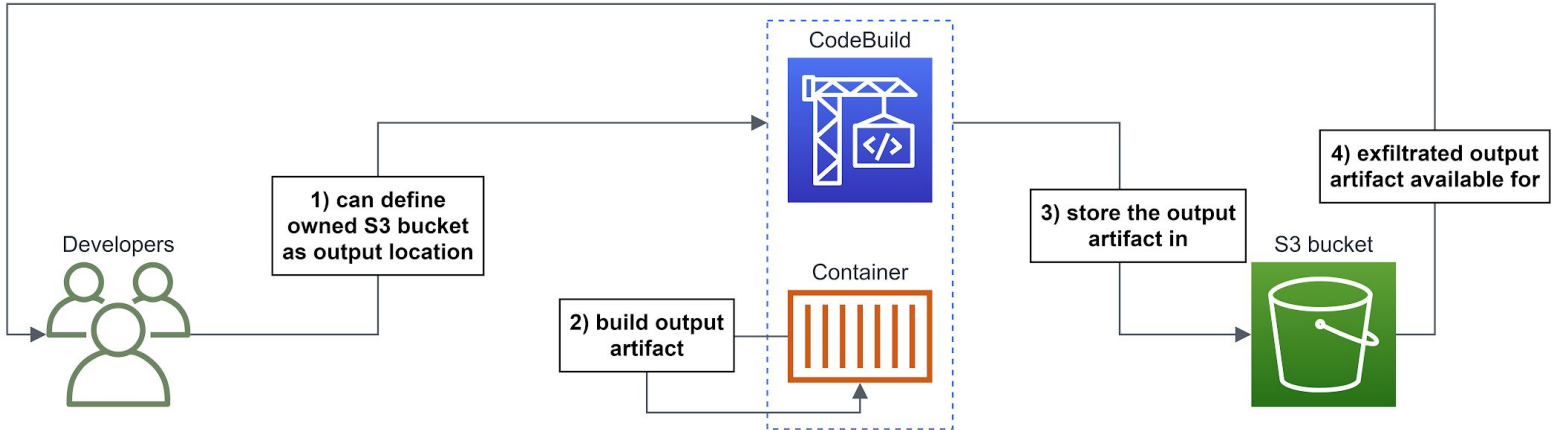
2

Tamper with the application

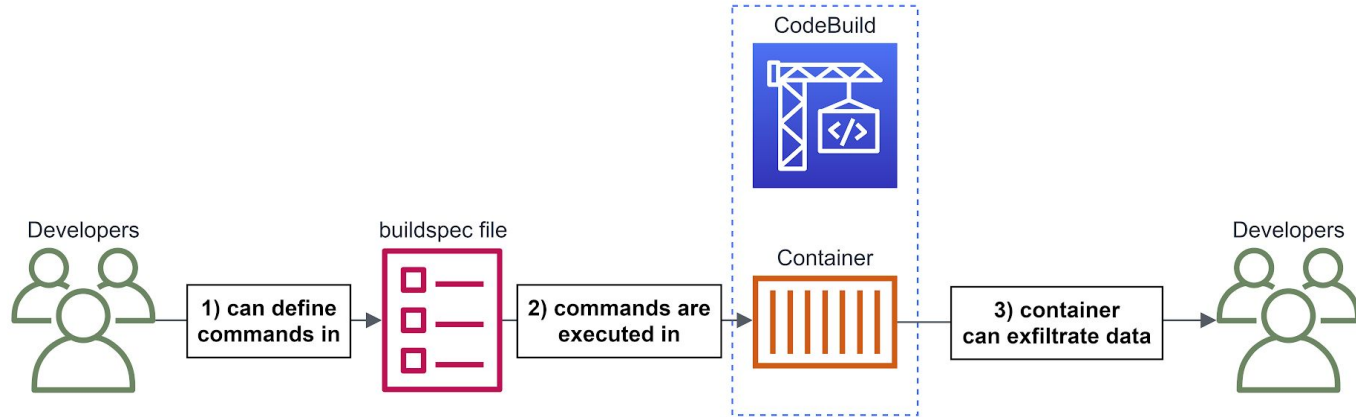
3

Run privileged commands in the  
deployment server

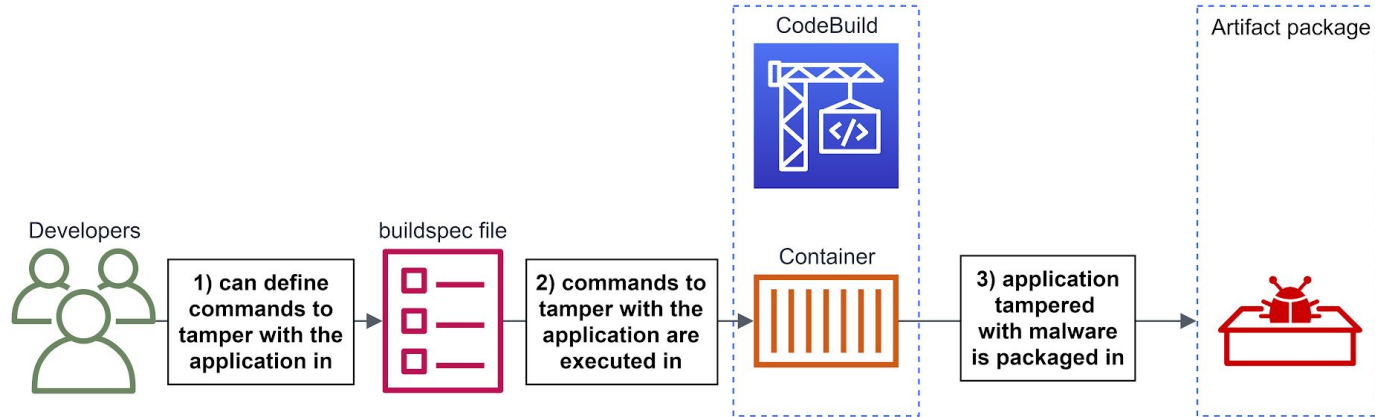
# 1. Exfiltrate sensitive data (Option 1)



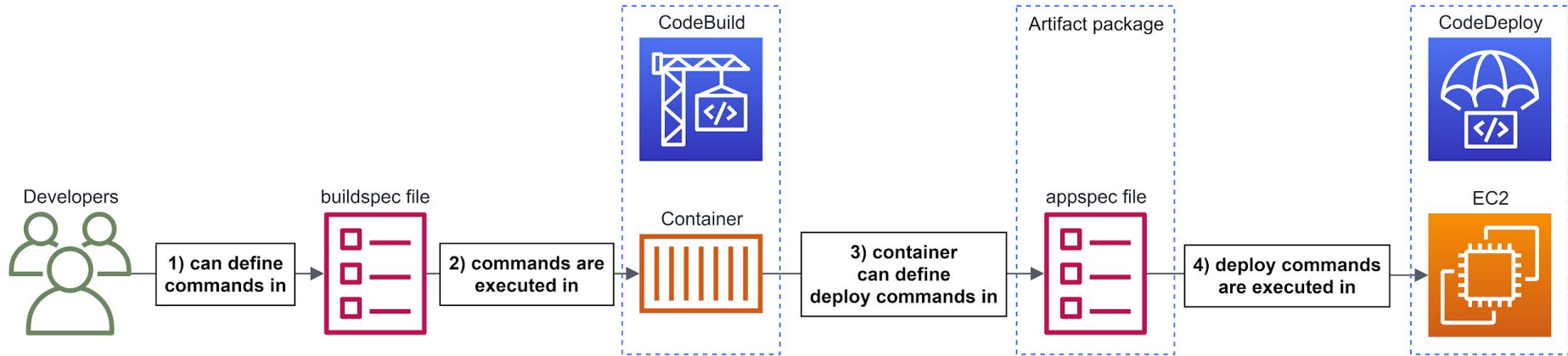
# 1. Exfiltrate sensitive data (Option 2)



## 2. Tamper with the application

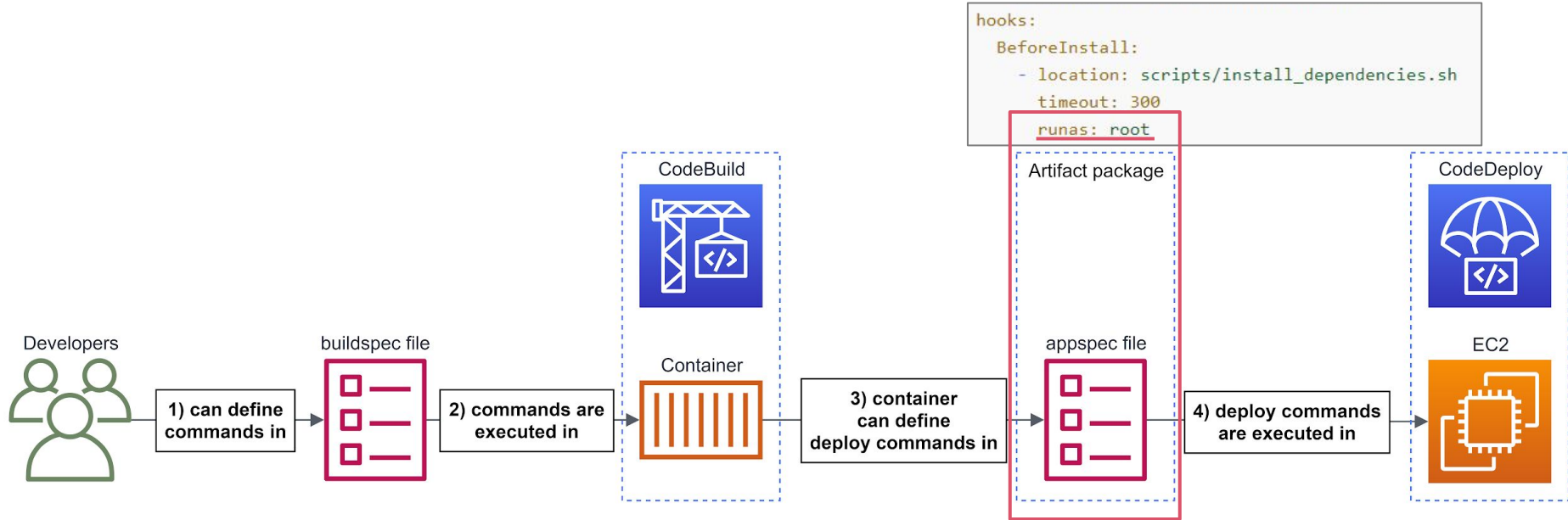


### 3. Run privileged commands in the deployment server

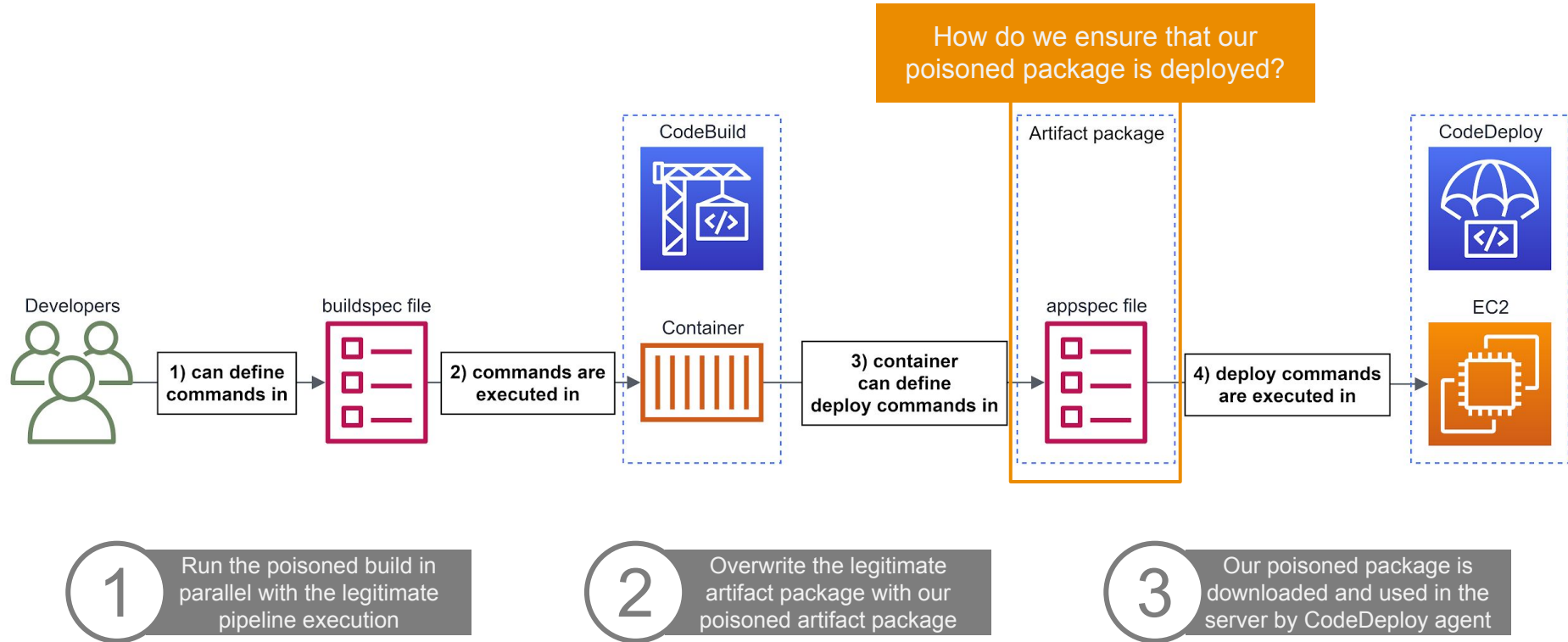




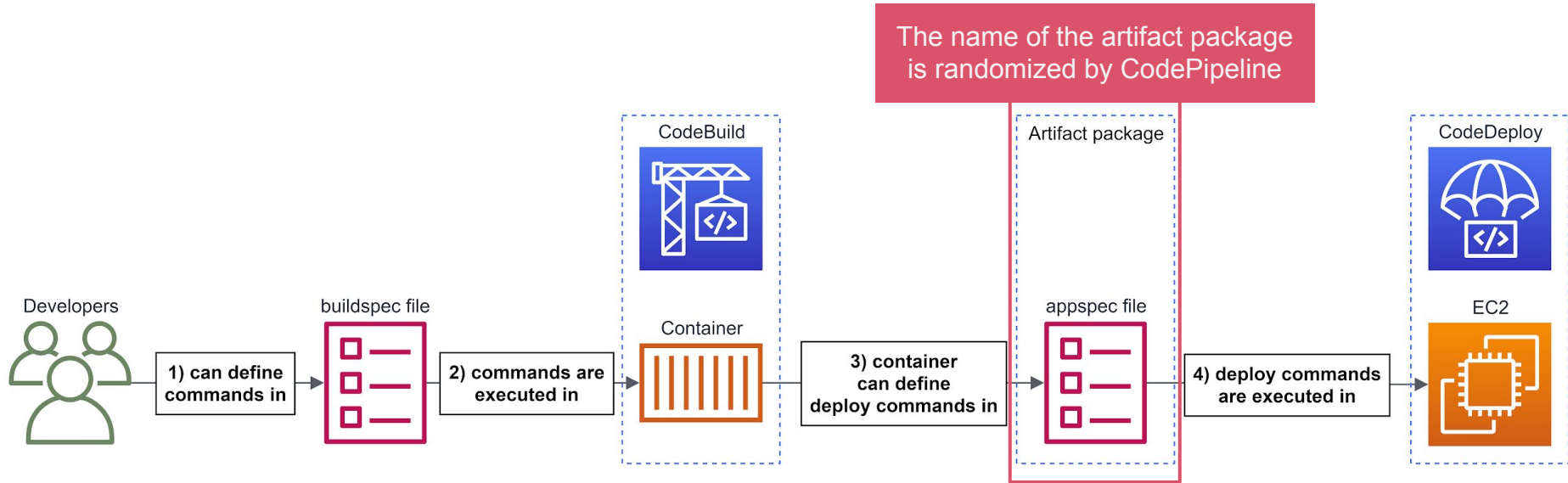
### 3. Run privileged commands in the deployment server



### 3. Run privileged commands in the deployment server



### 3. Run privileged commands in the deployment server



# Find the name and location of the legitimate artifact

## Option 1: CodePipeline actions

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "codepipeline:ListActionExecutions",
        "codebuild:StartBuild"
      ],
      "Resource": "*"
    }
  ]
}
```

Included in **CodePipeline Read Only** policy  
(may be provided to developers)

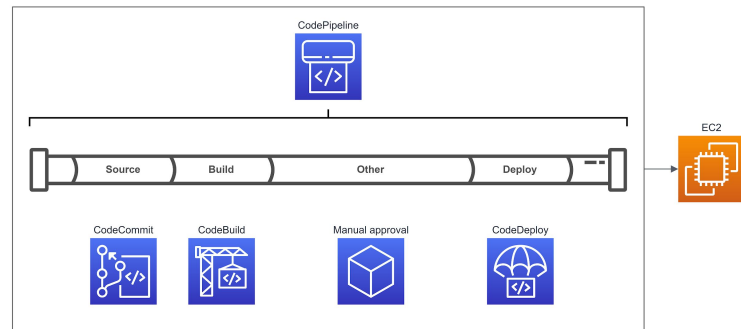
## Option 2: CodeBuild actions

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "codebuild:ListBuilds",
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
      ],
      "Resource": "*"
    }
  ]
}
```

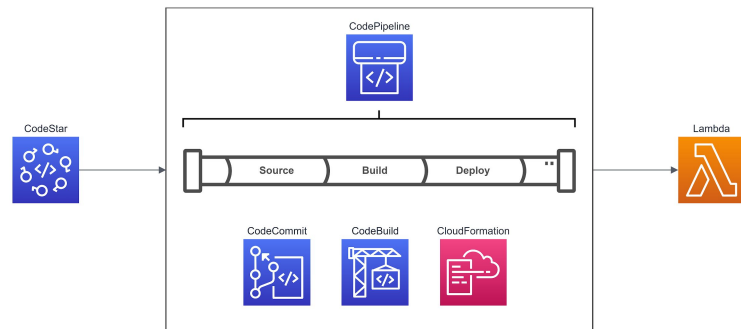
Included in **CodeBuild Developer** policy &  
**CodeStar Contributor** role

# Demo time

## Server deployment CI/CD pipeline

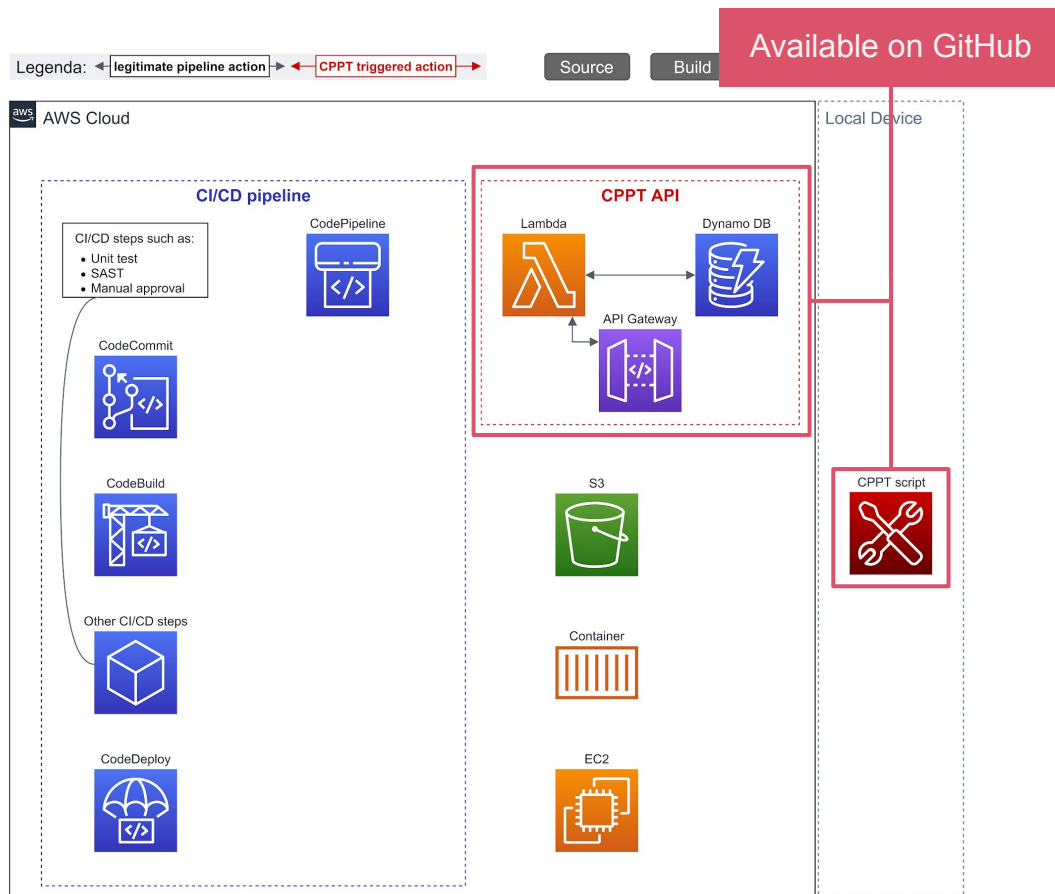


## Serverless deployment CI/CD pipeline



# Server deployment demo

Using the CodePipeline Poisoning Tester (CPPT) tool



<https://github.com/AsierRF/CodePipeline-Poisoning-Tester>

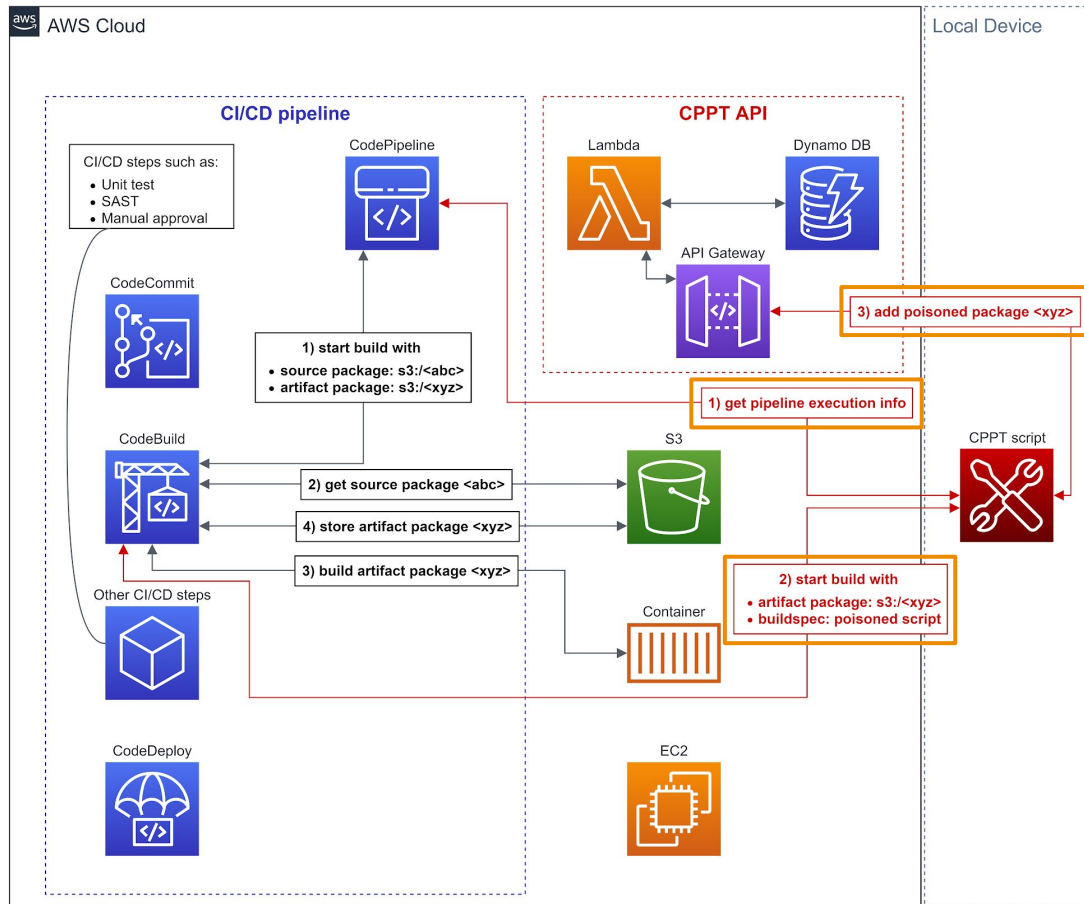
## Send HTTP request

### POST to CPPT API

- Monitoring the poisoning process
- Test internet access

1

Exfiltrate data



## Create a file

### CPPTWasHere.\*

- **Container:** Added to the artifact package
- **Server:** Created in the root (/) directory

2

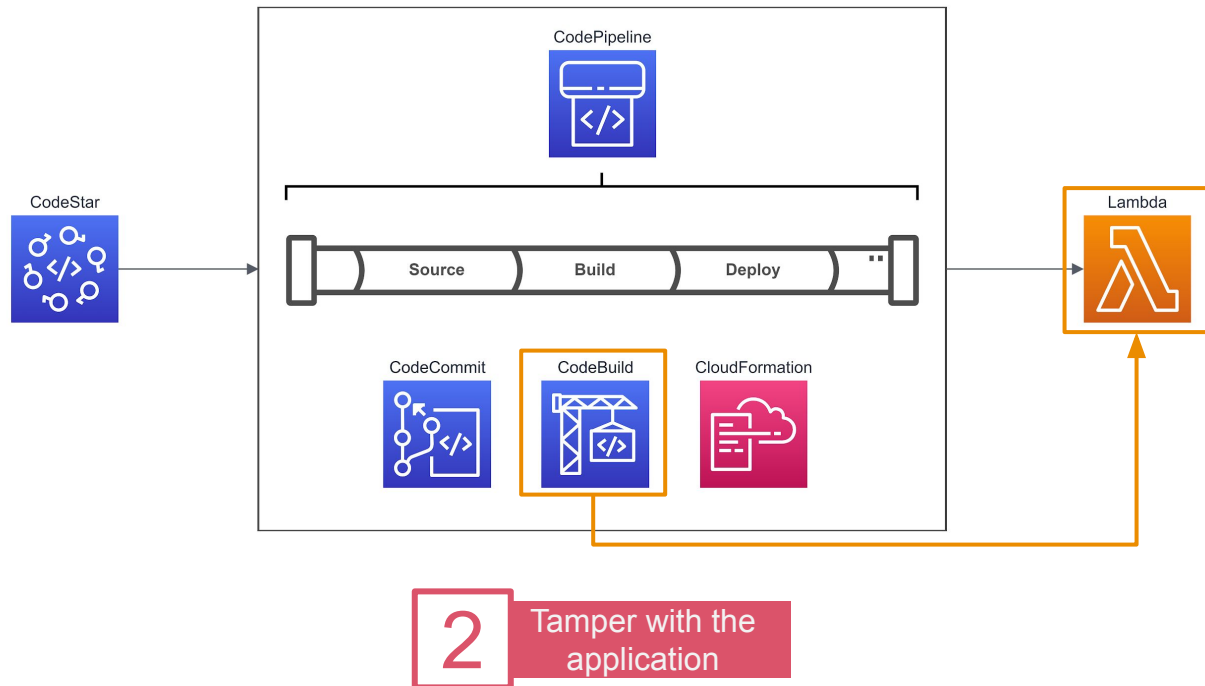
Tamper with the application

3

Run privileged commands

# Serverless deployment demo

## BruCON Web Application





# Are these risks relevant in the real world?

Is **exfiltration** really an issue since ...

a developer can read the source code & read the documentation?

Often, access is restricted to a part of the codebase, but CodeBuild has full access

Is **tampering** really an issue since ...

a developer can push code and functionalities?

Peer reviews help reduce the risk of rogue developers, but CodeBuild allows bypassing them

Is **execution** really an issue since ...

a developer can often run commands to configure & administer a server?

It is a known and accepted risk, but CodeBuild provides an unknown and shadow approach

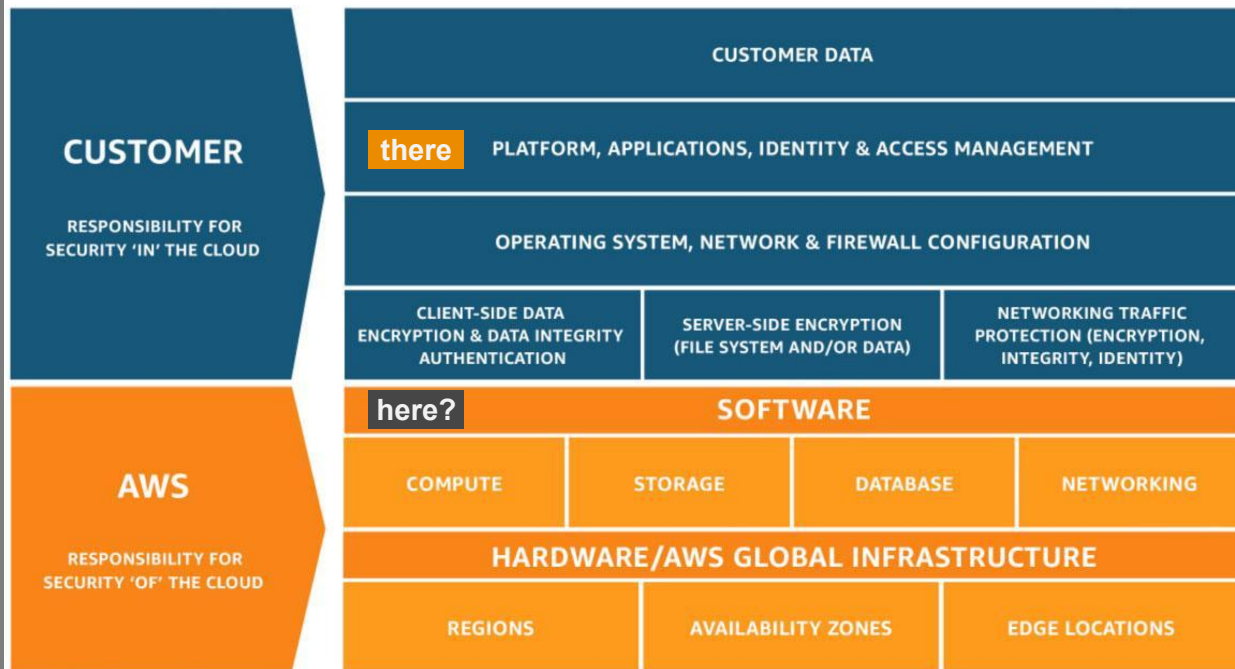
Also, ...

a developer is a trusted person in the company

Credentials can be leaked, developers can be external, blackmailing, other breaches, ...

# Who is responsible for these risks?

## Shared responsibility model



# I recommend you to review your CI/CD pipelines in AWS

If you **can deny** access to the  
StartBuild action

**Deny it**, automate the initialization of the build execution with hooks

If you **must allow** access to the  
StartBuild action

If users do **not require** the  
'override' parameters

Leverage **Lambda**, users can trigger a lambda function, which  
starts a build on behalf of the user without the 'override' parameters

If users **require** the  
'override' parameters

Rethink your pipeline:

- **Identify and evaluate** sensitive tasks
- **Split and order** the tasks
- **Deny** access to, at least, sensitive tasks

# Detailed information in our publication

<https://pwc.to/2VYrpZK>

## The risk of CI/CD pipeline poisoning via CodeBuild

On the intricate challenges of setting up a secure

### CodeBuild

The CodeBuild service offers a simple solution to configure, provision, manage and scale-build servers and environments. The computing power is supported by container instances that focus on running jobs, such as code compilation, code testing, and code analysis. The key features of CodeBuild are the simplicity of integration in existing and new CI/CD pipelines and the flexibility provided through an AWS managed service. CodeBuild is a core service for AWS based pipelines, but it also relies on the intervention of other AWS services, such as Elastic Container Service (ECS) and Elastic Container Registry (ECR) for the provision of computational power via container instances, Simple Storage Service (S3) for the storage of source code packages and applications, and Secrets Manager for the access to passwords and parameters.

In order to perform its service, the CodeBuild container instance has access to resources. This is achieved by providing the access rights to the source storage services, which can be version control repositories, such as CodeCommit or GitHub, or version control repositories, such as CodeCommit or GitHub, or access rights can be provided via an IAM role for AWS via access credentials for non-AWS service providers. This is a so-called service role, is applied to the container instance, configured at CodeBuild project level. The access credentials are configured at project level; however, these are defined in the source service configuration. The access credentials are in the form of an access token or OAuth credentials and are managed by CodeBuild within AWS.

Every time a CodeBuild build execution is started, CodeBuild ECS to deploy a new container instance with a container image configured at project level. ECS will then set up and run the container with the target container image, which can be provided by ECR service or another non-AWS container image repository. Once the container is ready to be used, the target source code will be downloaded into the container instance, which will then execute the resources required to perform the job. These resources include files, compiled libraries, applications and others. In our publication, we named this process the **Provision Task**.

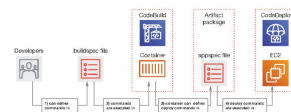
PwC | 5

### Execution of privileged commands in deployment server

A developer who can start a build execution can potentially run arbitrary privileged commands in the deployment server if the CodeBuild project is used as part of a CI/CD pipeline, for example, orchestrated by CodePipeline.

The `buildspecOverride` parameter can be used by a developer to specify the commands executed by the container instance of CodeBuild. These commands can be used to incorporate new scripts and modify the scripts included in the artifact package. These scripts are likely to be executed in the deployment server during the installation of the new version via the artifact package.

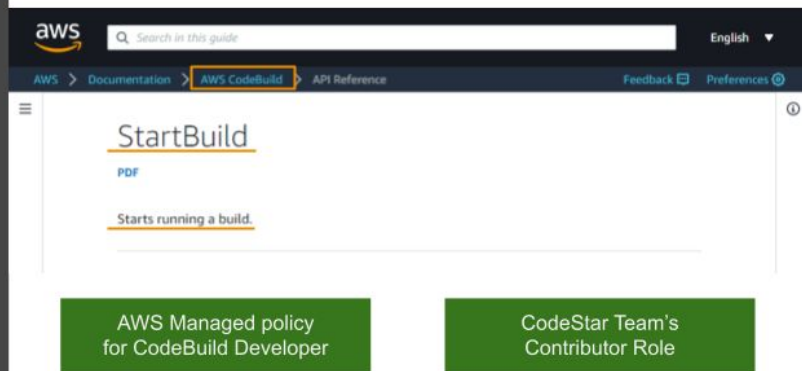
In CI/CD pipelines based on AWS services, the artifact package will include the AppSpec file used by the CodeDeploy Agent. Therefore, the AppSpec file can be altered within the CodeBuild container instance via the commands included in the BuildSpec file. In addition, the `runas` property of the AppSpec file allows the user to determine that the arbitrary commands and scripts are meant to be run as a privileged user (i.e. root) without any password or authentication being required to elevate the user. This can provide the developer full control over the system if the artifact is installed. The following image illustrates this attack vector.

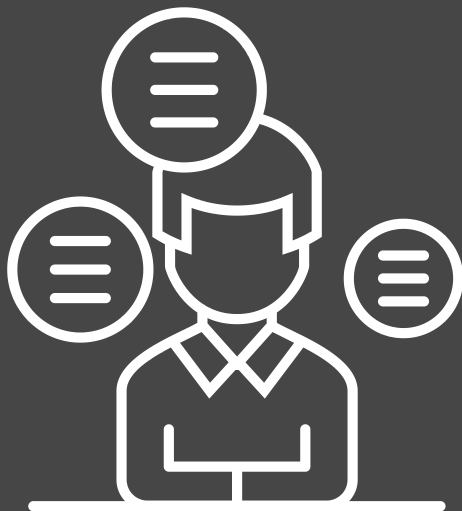


PwC | 10

Do we  
understand  
cloud services  
well enough?

Would you still  
allow me to ...





Looking forward to  
your comments and  
questions