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?

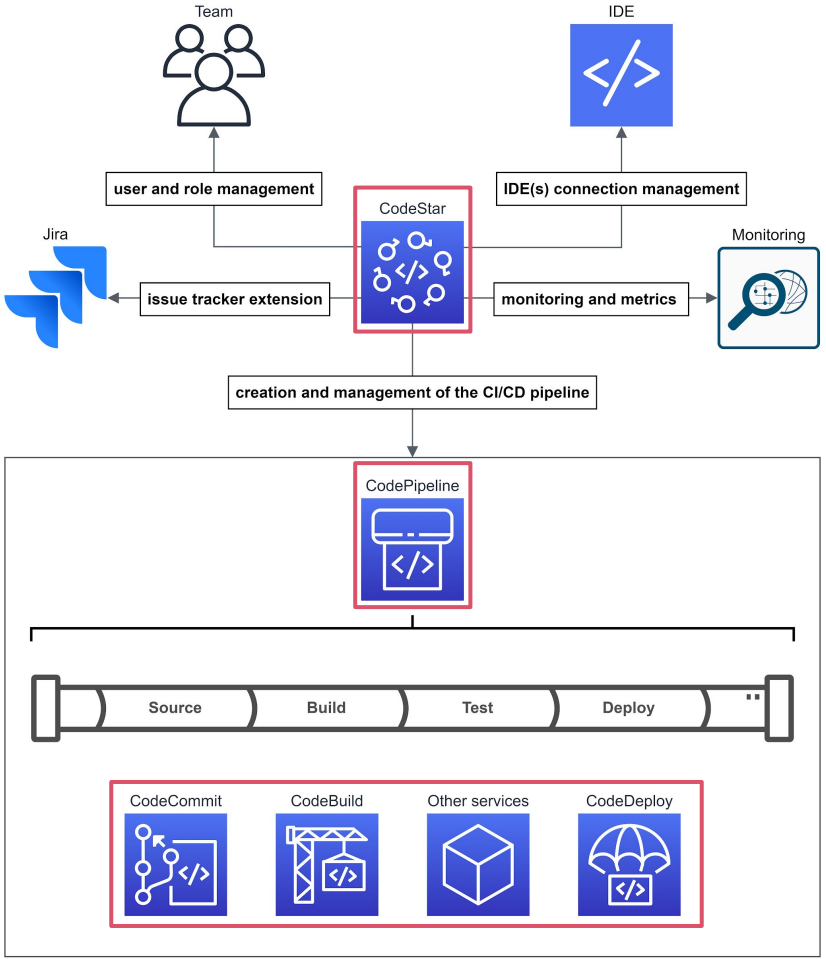
Deep dive into CI/CD pipelines in the cloud

Risk analysis

Threat modelling

Tailor CI/CD pipeline security in the cloud

CI/CD pipelines in AWS, the Code services



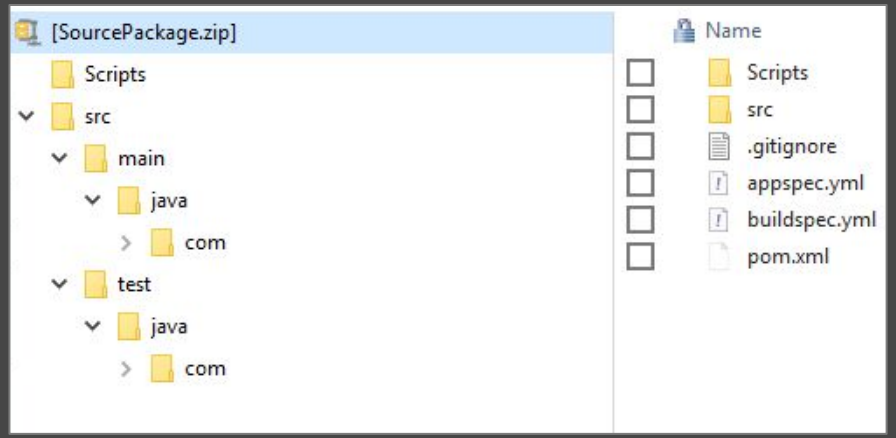
source package

Source package

Created by CodeCommit / Used by CodeBuild

Resources required to compile and configure the application:

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



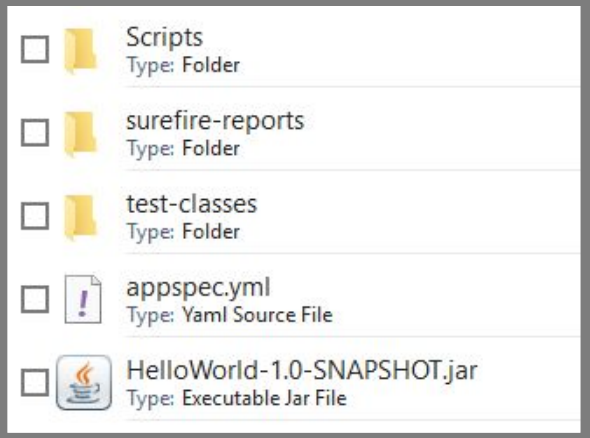
artifact package

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

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

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:

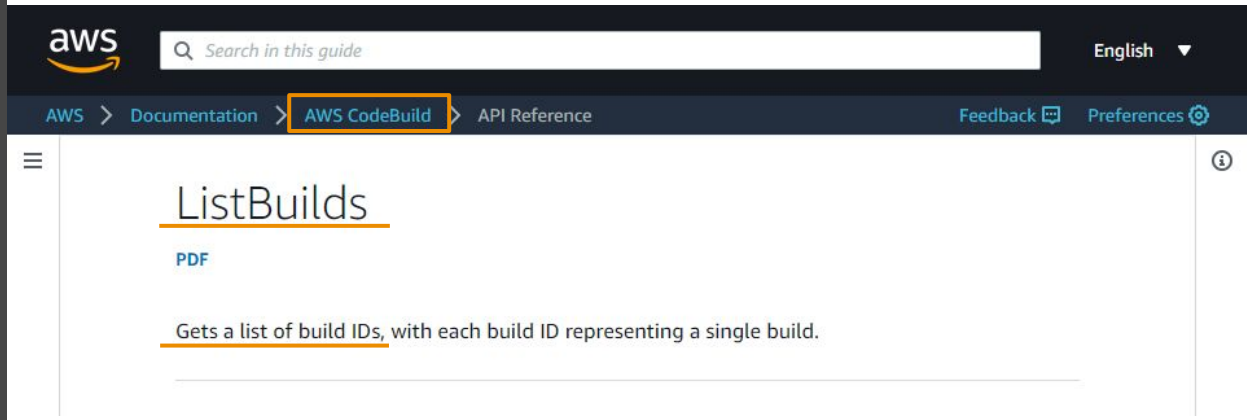
- **Asier: new developer** at the company, I need access rights.
- **Audience: the administrator** and you have been requested to provide the developer with the correct access rights.

AWS infrastructure:

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

AWS Managed policies	AWS CodeStar Team roles
AWSCodeBuild**Admin**Access	**Owner**
AWSCodeBuild**Developer**Access	**Contributor**
AWSCodeBuild**ReadOnly**Access	**Viewer**

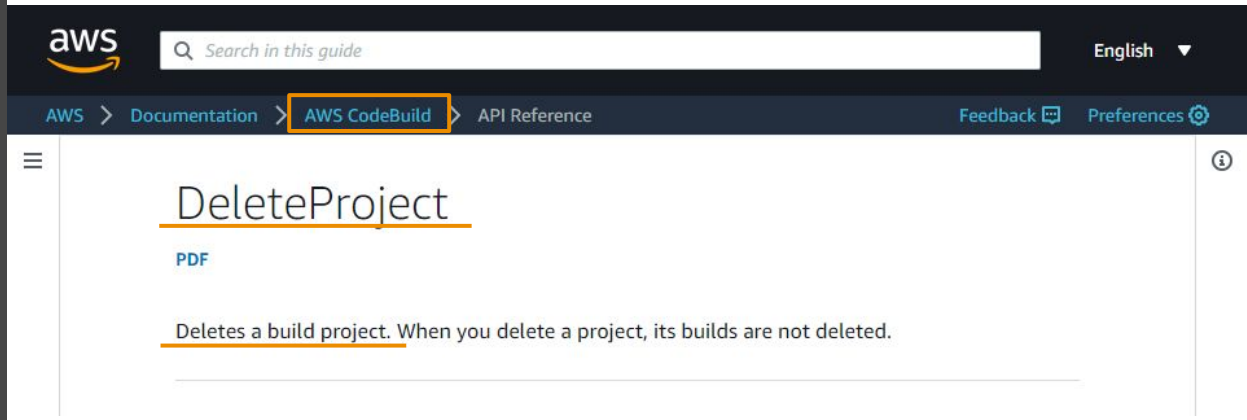
Would you allow me to …



AWS Managed policy for CodeBuild Developer

CodeStar Team's Contributor Role

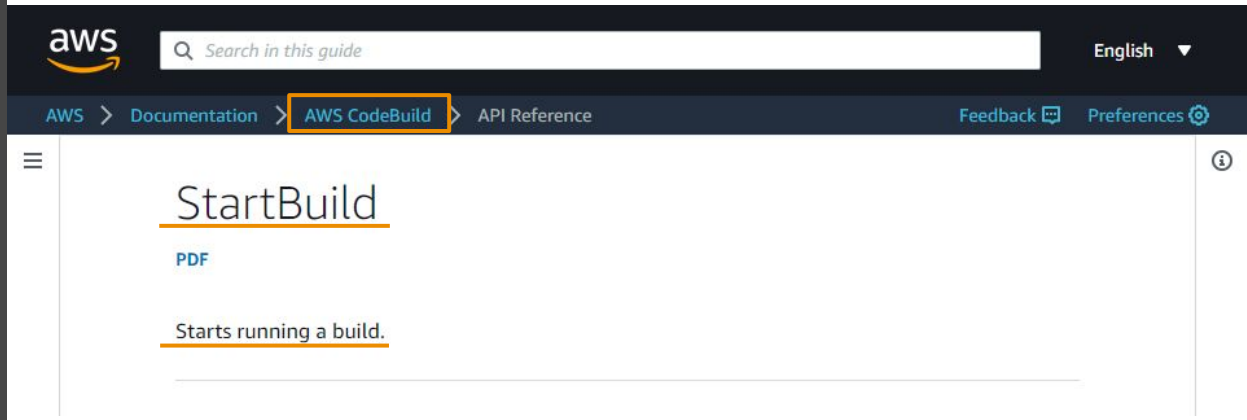
Would you allow me to …



AWS Managed policy for CodeBuild Developer

CodeStar Team's Contributor Role

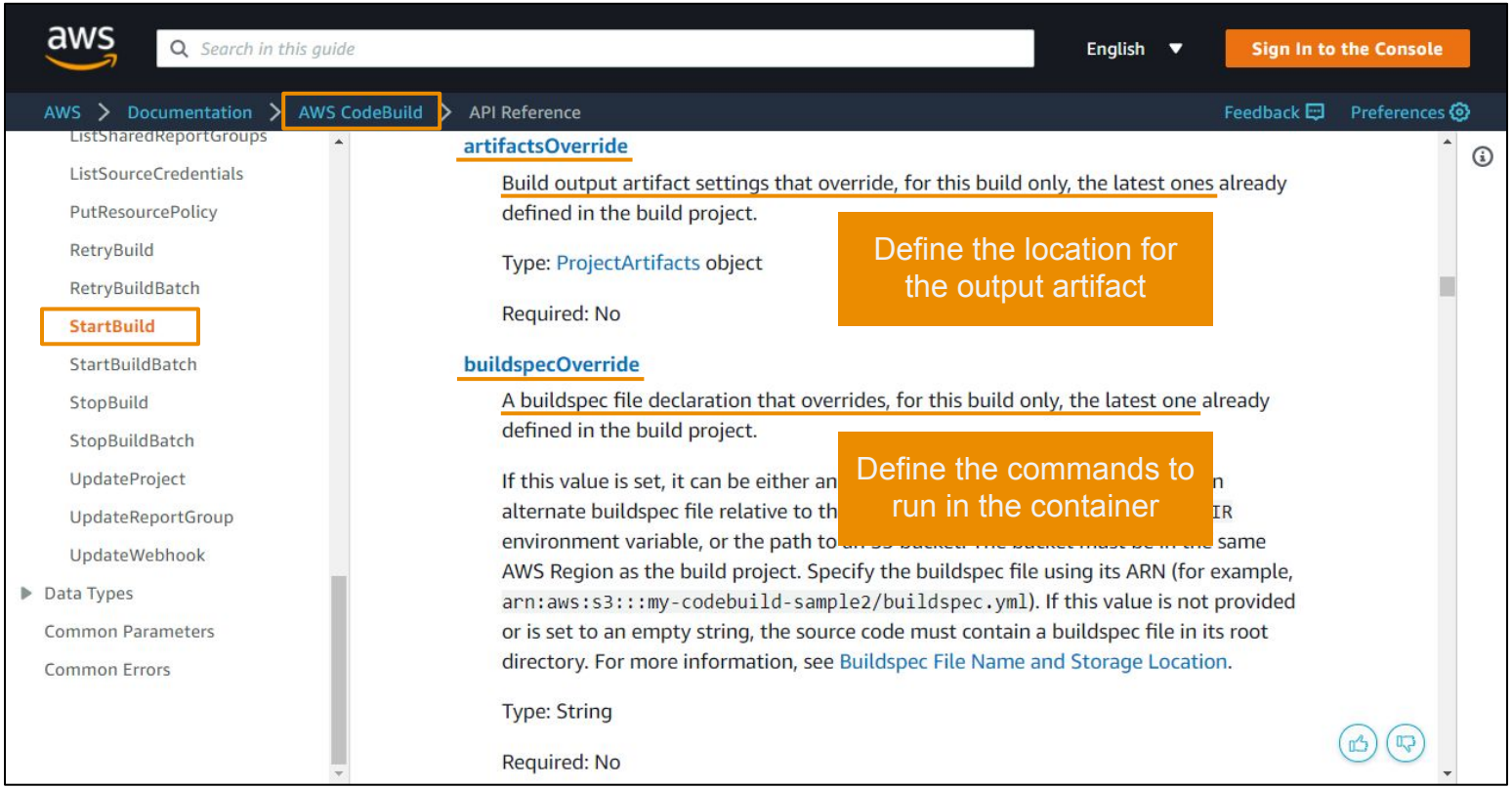
Would you allow me to …



AWS Managed policy for CodeBuild Developer

CodeStar Team's Contributor Role

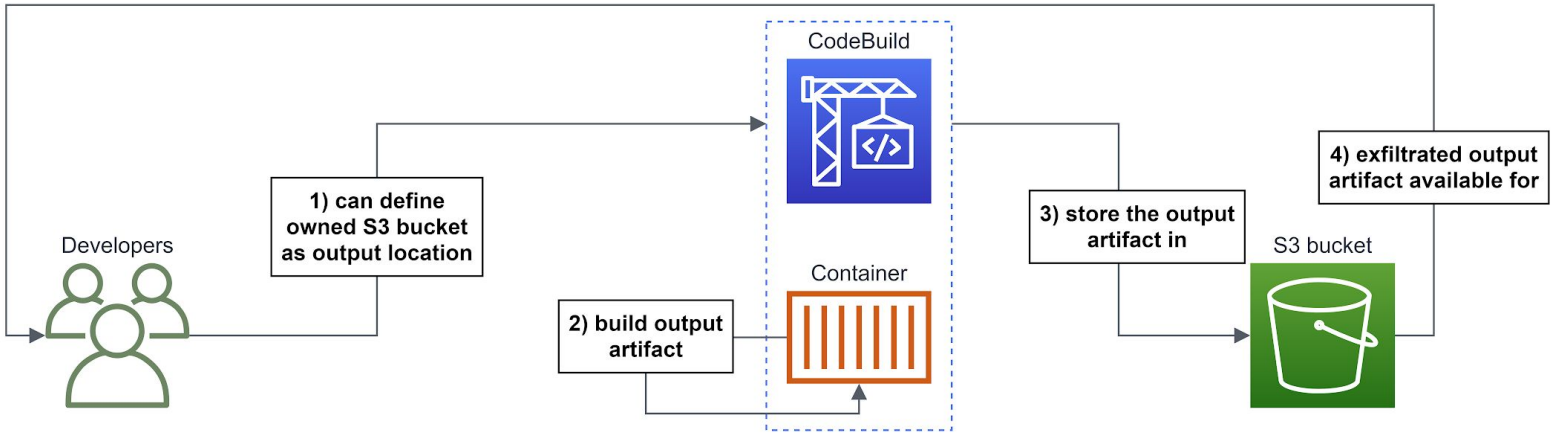
StartBuild parameters



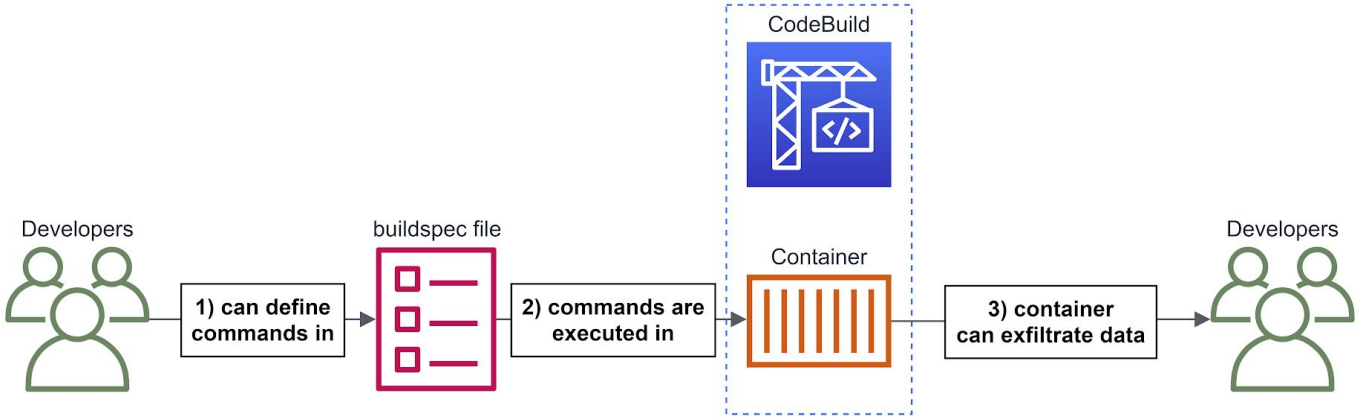
An AWS developer can abuse StartBuild to ...

1. Exfiltrate data
2. Tamper with the application
3. Run privileged commands in the deployment server

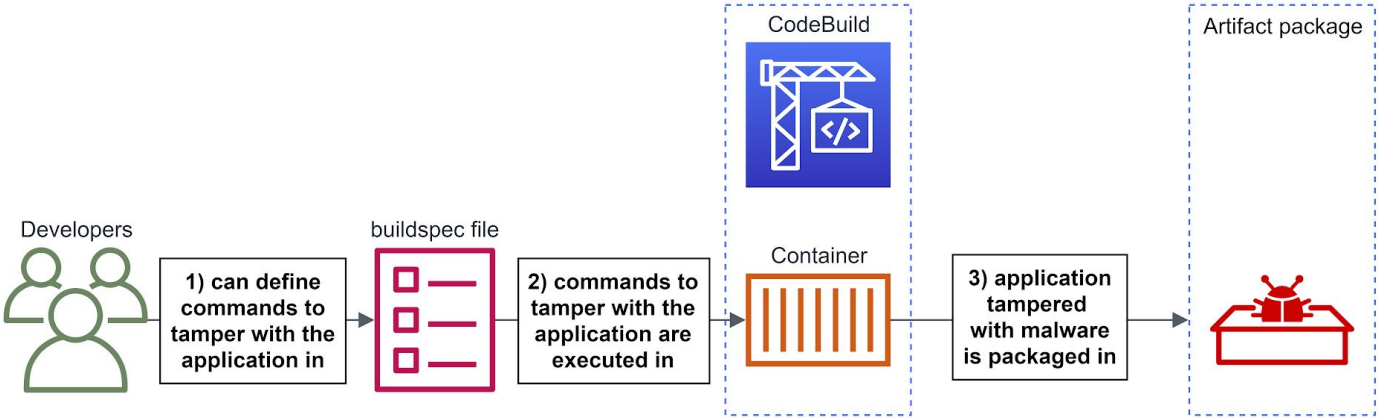
1. Exfiltrate sensitive data (Option 1)



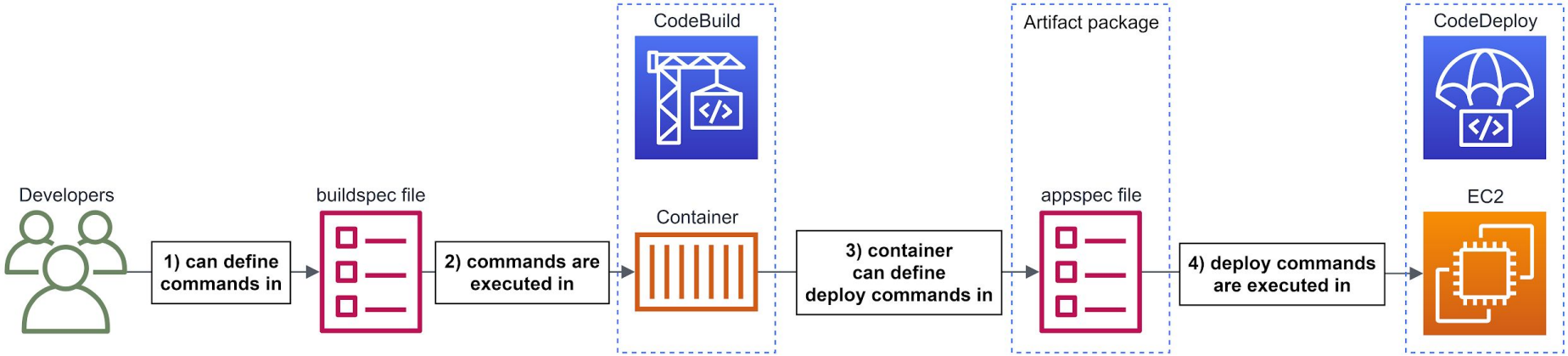
1. Exfiltrate sensitive data (Option 2)



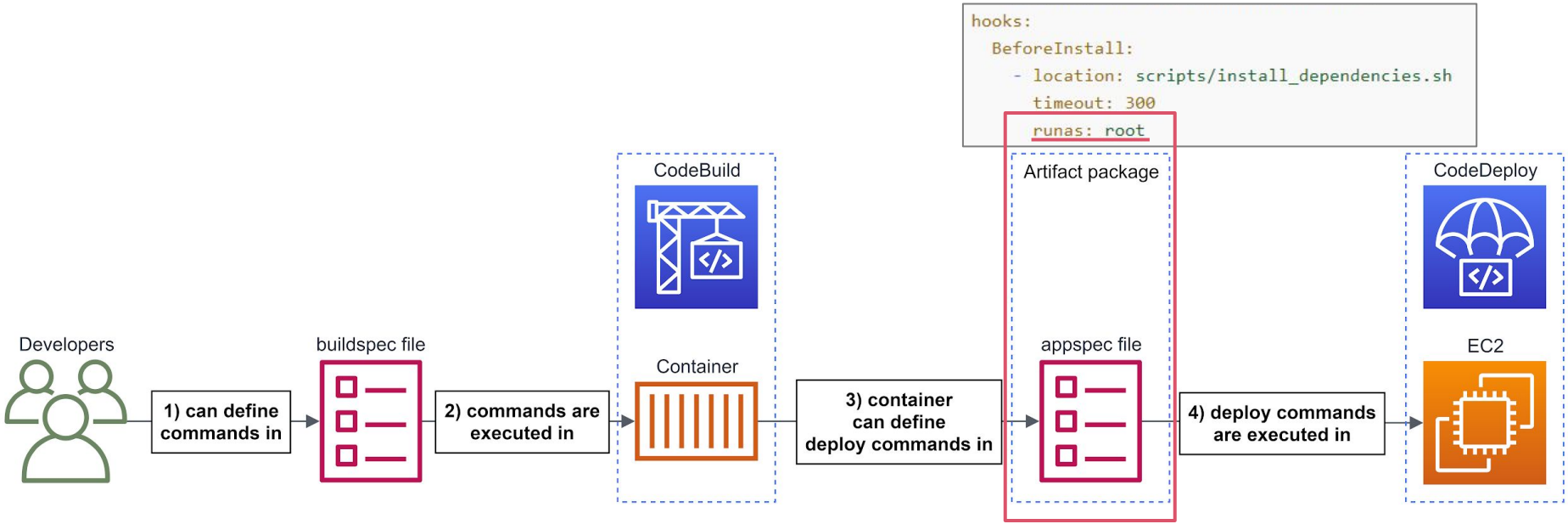
2. Tamper with the application



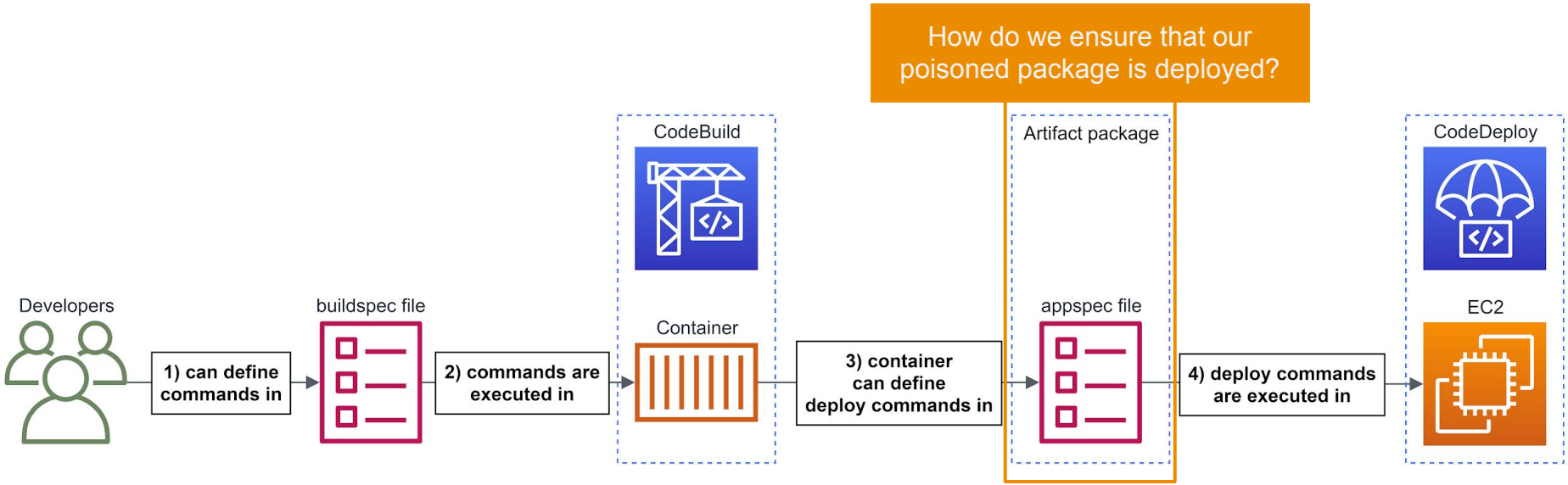
3. Run privileged commands in the deployment server



3. Run privileged commands in the deployment server

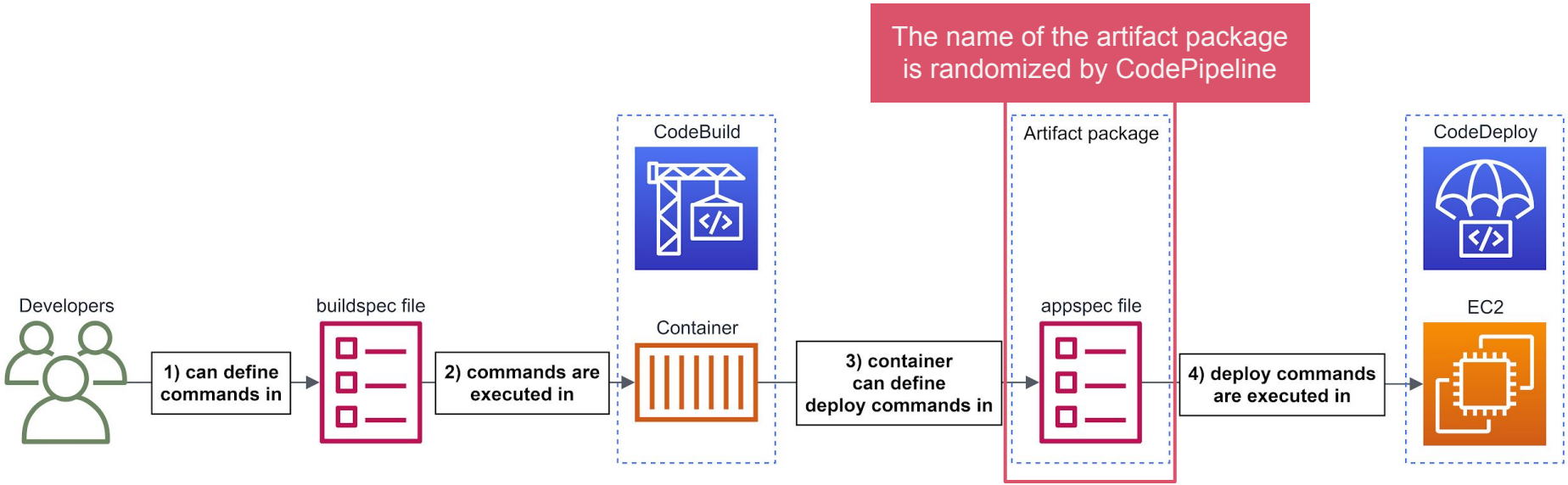


3. Run privileged commands in the deployment server



1. Run the poisoned build in parallel with the legitimate pipeline execution
2. Overwrite the legitimate artifact package with our poisoned artifact package
3. Our poisoned package is downloaded and used in the server by CodeDeploy agent

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)

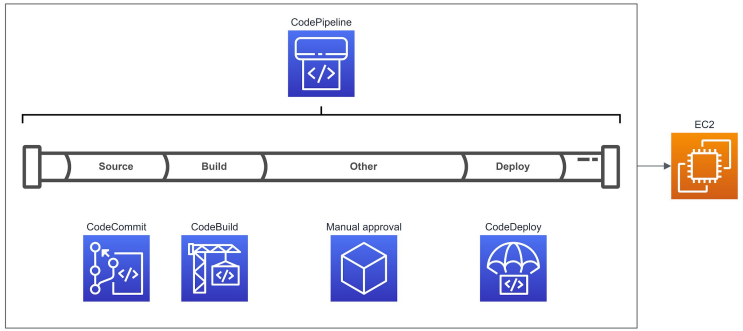
Option 2: CodeBuild actions

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

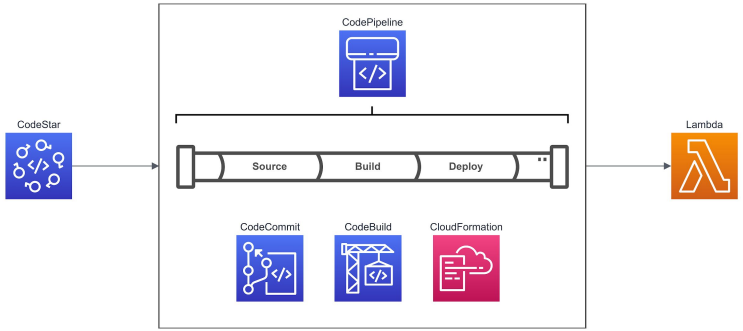
Included in **CodeBuild Developer policy** &
CodeStar Contributor role

Demo time

Server deployment
CI/CD pipeline

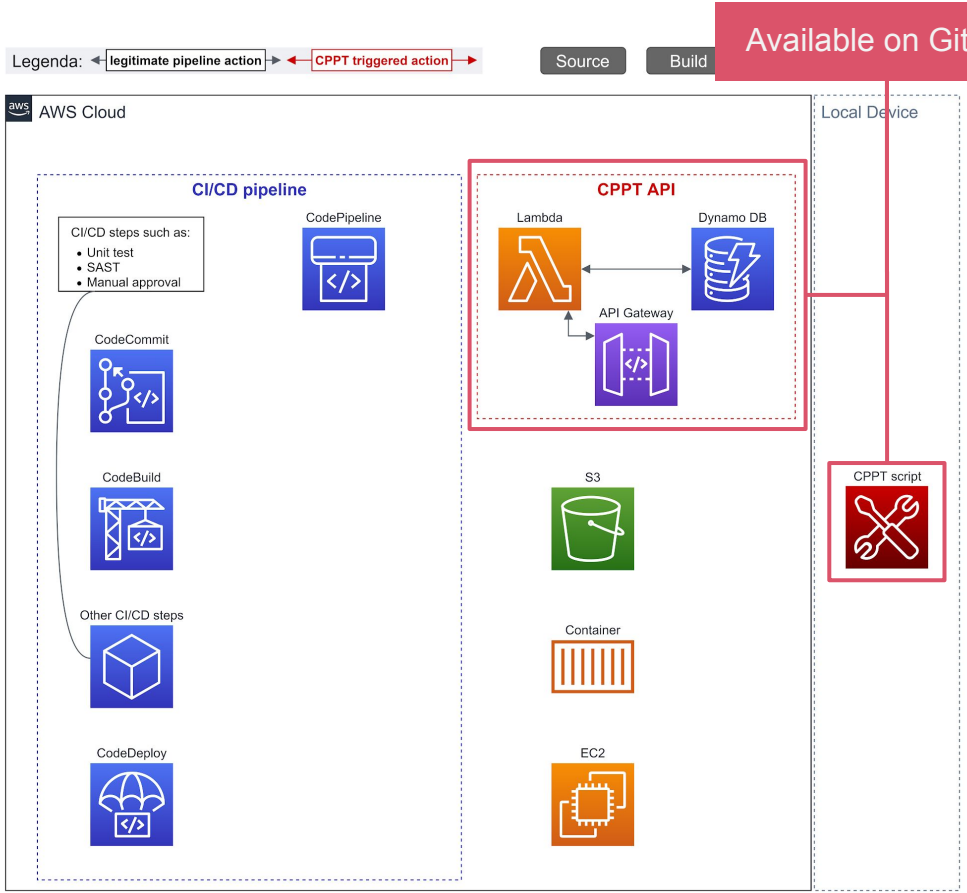


Serverless deployment
CI/CD pipeline

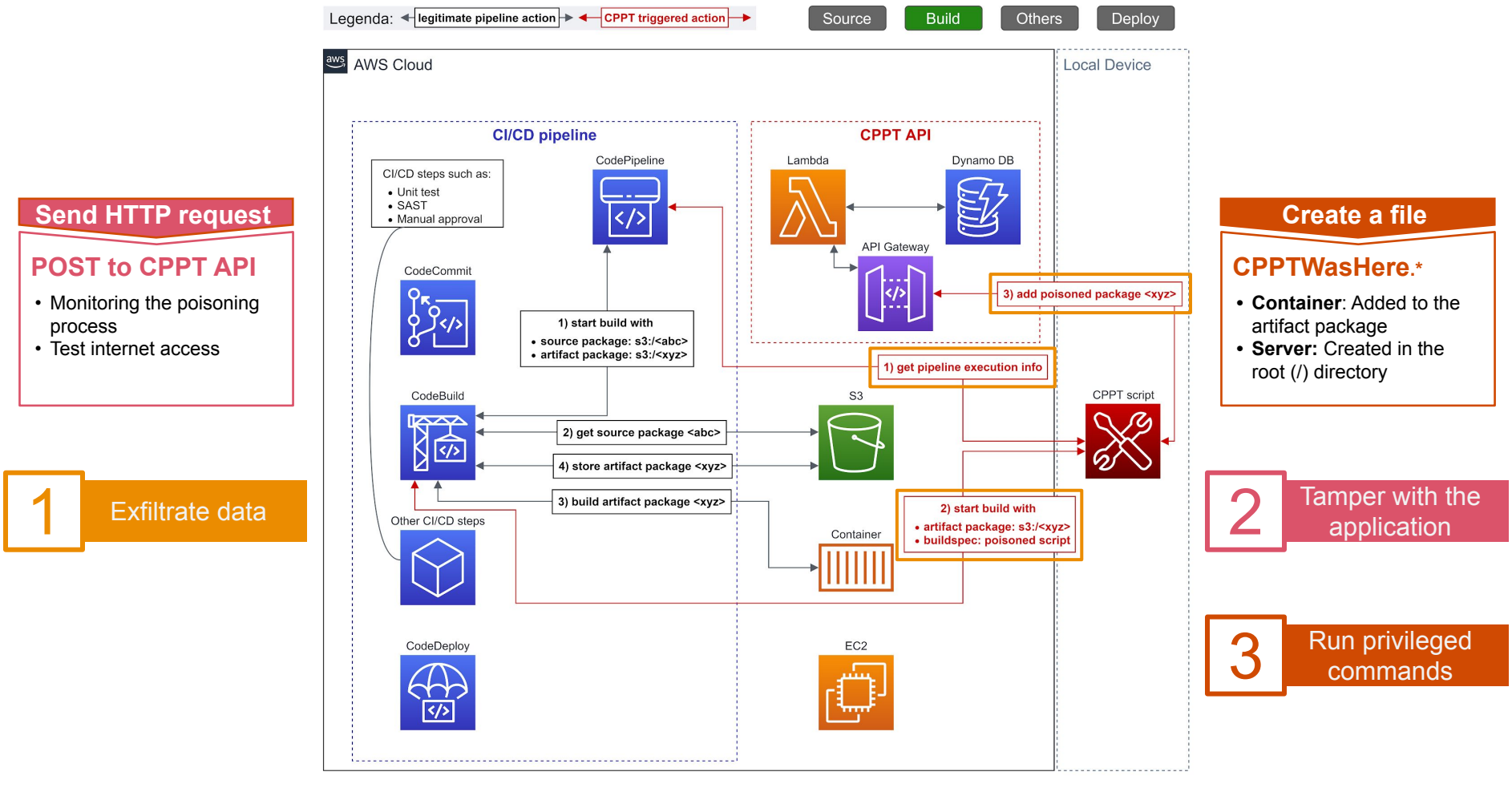


Server deployment demo

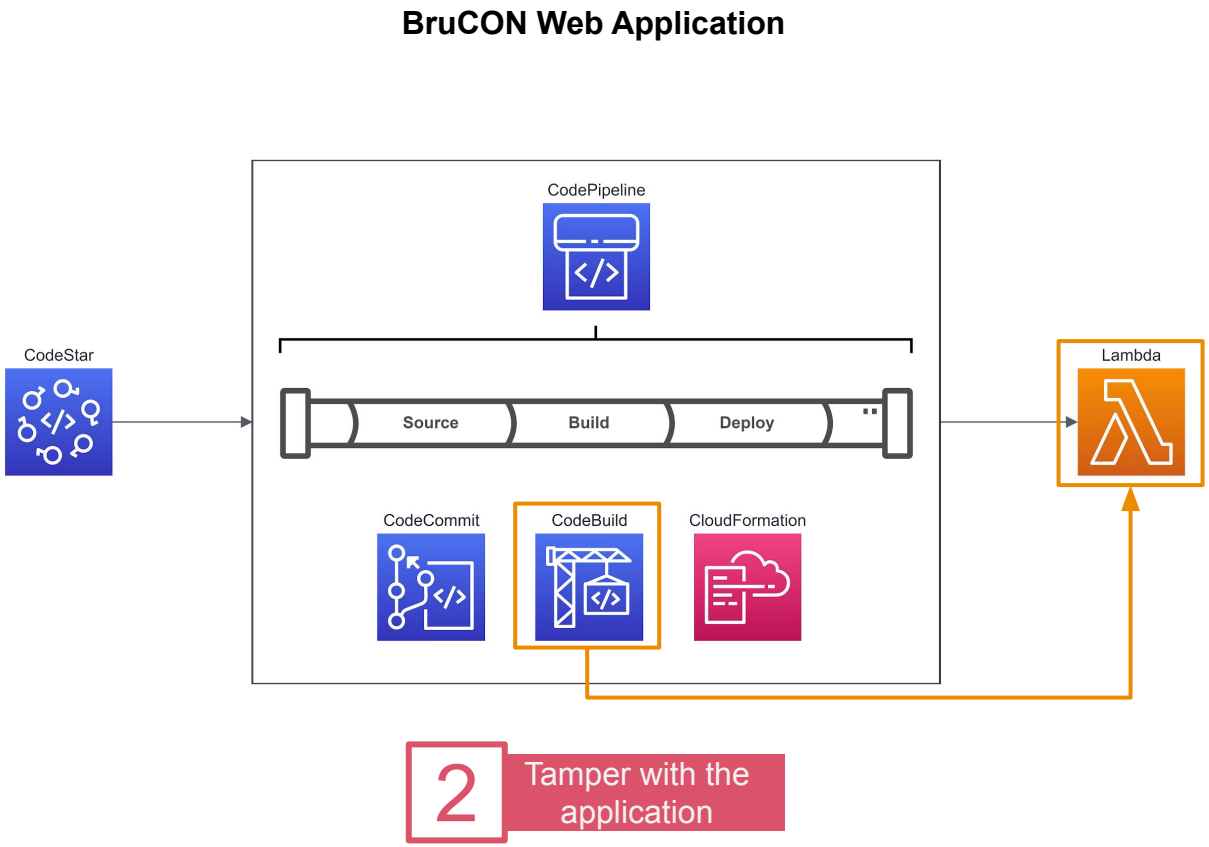
Using the CodePipeline Poisoning Tester (CPPT) tool



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


Legenda:
legitimate pipeline action
CPPT triggered action
Source
Build
Others
Deploy
AWS Cloud
Local Device
CI/CD pipeline
CPPT API
CI/CD steps such as:
• Unit test
• SAST
• Manual approval
CodePipeline
Lambda
Dynamo DB
API Gateway
CodeCommit
3) add poisoned package <xyz>
1) start build with
• source package: s3:/<abc>
• artifact package: s3:/<xyz>
1) get pipeline execution info
CodeBuild
S3
CPPT script
2) get source package <abc>
4) store artifact package <xyz>
3) build artifact package <xyz>
2) start build with
• artifact package: s3:/<xyz>
• buildspec: poisoned script
Other CI/CD steps
Container
CodeDeploy
EC2
Send HTTP request
POST to CPPT API
• Monitoring the poisoning process
• Test internet access
Create a file
CPPTWasHere.*
• Container: Added to the artifact package
• Server: Created in the root (/) directory
1 Exfiltrate data
2 Tamper with the application
3 Run privileged commands

Serverless deployment demo



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

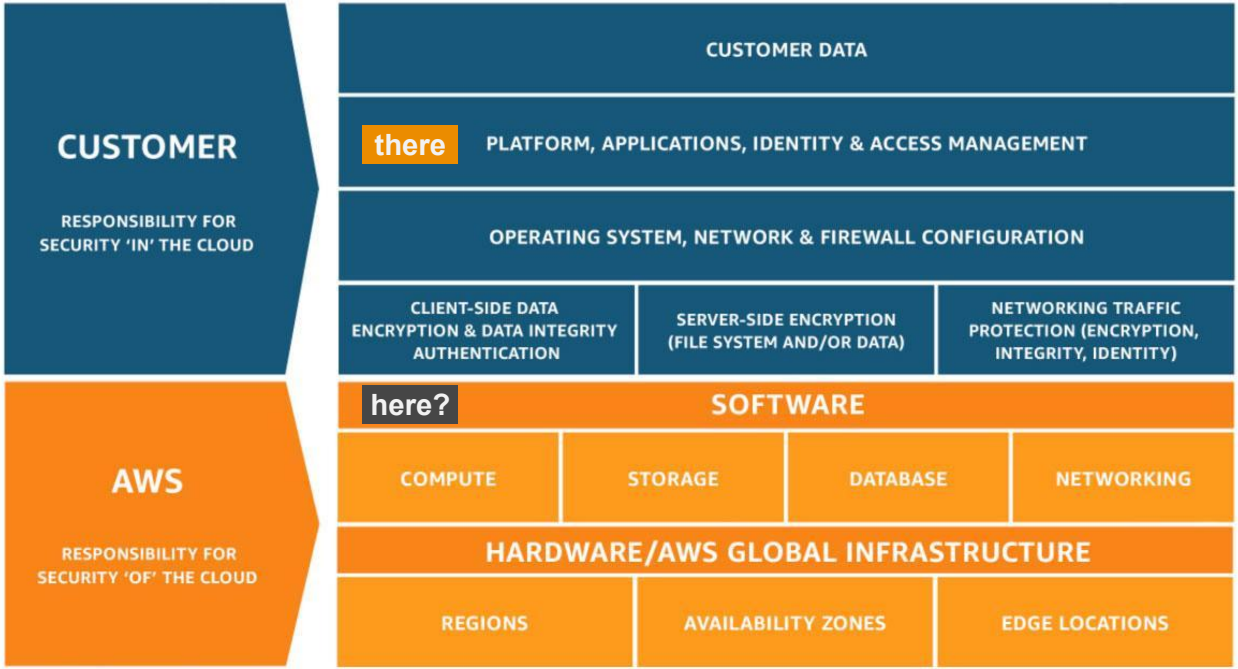
Also, ...

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

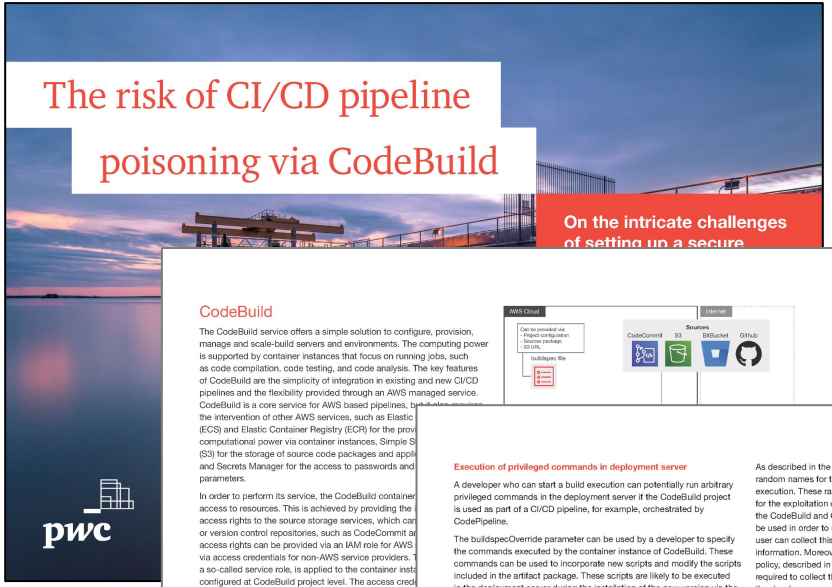
If users **require** the ‘override’ parameters

Rethink your pipeline:

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

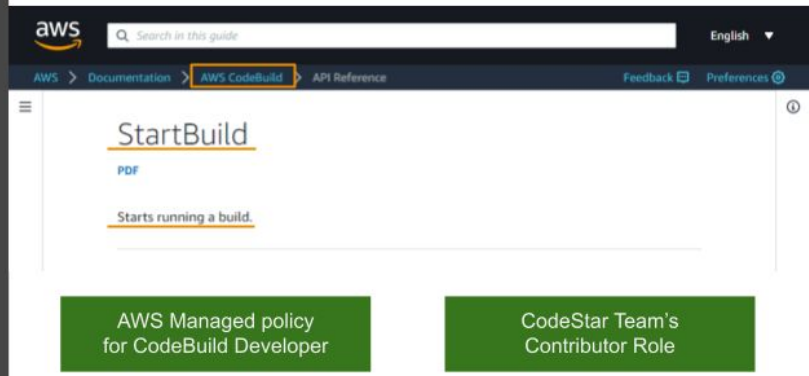
Detailed information in our publication

https://pwc.to/2VYrpZK



Do we understand cloud services well enough?

Would you still allow me to ...



Looking forward to your comments and questions