



How to automate the Development Process of OpenEdge[®] on

By Paul Guggenheim

About PGA

- OpenEdge Evangelist, Developer, Trainer, and Course Designer

- **Progress/OpenEdge** Partner



- Amazon Authorized Instructor



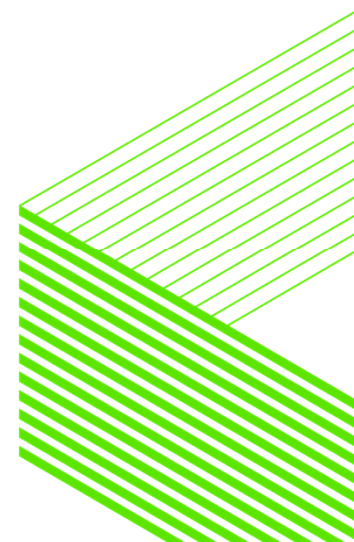
- AWS Solutions Architect



- **White Star** Software Strategic Partner

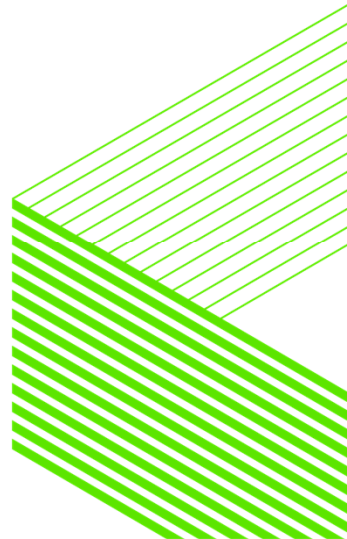
- **Consultingwerk** Partner

- **AppPro** Reseller



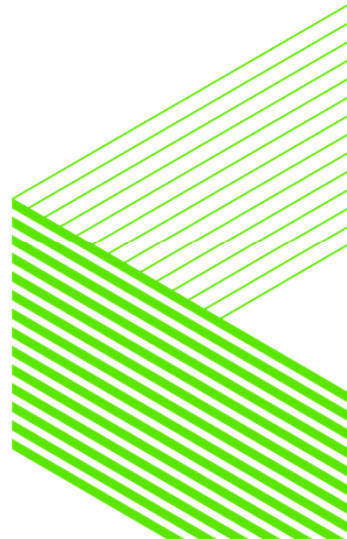
Overview

- What are the development tasks?
- How can we automate these development tasks?
- EC2 Image Builder
 - Purpose
 - Resources
 - Image Workflows
 - Output Images
 - Automate through AWS CLI
 - Develop Custom Components
 - Demonstration
- Cloud Formation
 - Infrastructure as Code
 - Demonstration



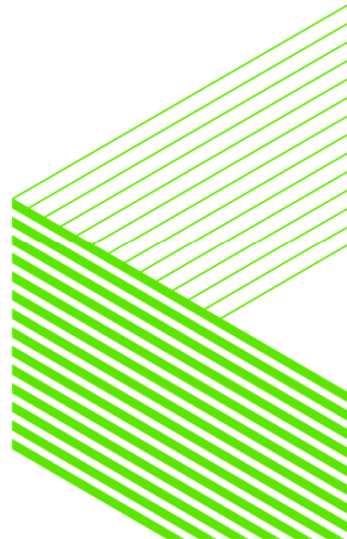
What are the development tasks?

- Create a development environment
 - Provision servers
 - Load software
 - Java
 - OpenEdge
 - Git
 - Kafka
 - Create Databases
 - Install source code
 - Configure Source Code Control System



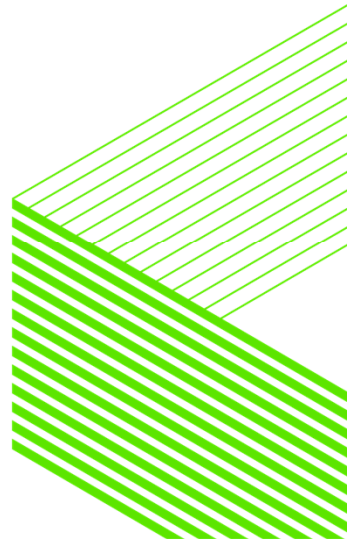
How can we automate these development tasks?

- Perform these tasks manually first
- Create Batch Scripts to automate these tasks
- Test the scripts on a local system if available
- Migrate resources and scripts to AWS Cloud
- Use EC2 Image Builder to build development images
- Apply Cloud Formation instructions to build instances from development images



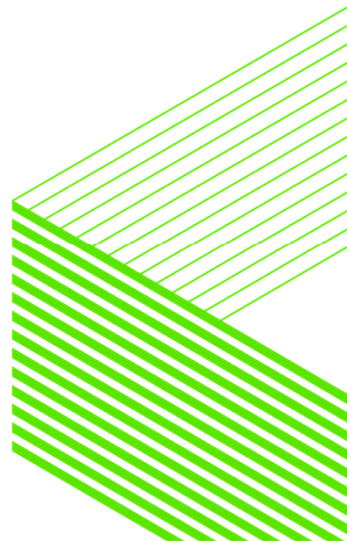
What is EC2 Image Builder?

- Fully-managed AWS Service
- Automate the creation, management and deployment of custom server images
 - Images meet specific IT Standards
 - Security
 - Easy to use environment
 - Provides for all required software to be loaded
 - Allows for efficient development and testing
 - Produces production images for scaling, automatic updates and patching



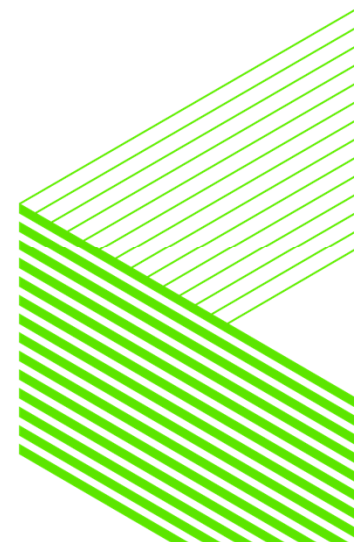
EC2 Image Builder Features

- Increase productivity and reduce operations for building compliant and up-to-date images
- Increase service uptime by testing images before deployment
- Creates images that remove unnecessary exposure to component security vulnerabilities
- Provides for centralized enforcement of policies to run instances from approved AMIs
- Simplified sharing of resources across AWS accounts
 - Components
 - Amazon-managed
 - Owned by me
 - Images
 - Image Recipes
 - Container Recipes



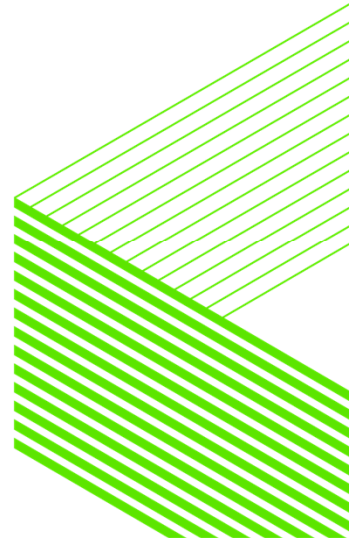
EC2 Image Builder

- Image Builder pipeline wizard guides you to create a custom image:
 - Choose a base image
 - Add or remove software from the base image
 - Customize settings and scripts with build components
 - Run selected tests or create custom test components to verify the image
 - Distribute AMIs to AWS regions and AWS accounts



EC2 Image Builder Supported Operation Systems

Operating system/distribution	Supported versions
Amazon Linux	2 and 2023
CentOS	7 and 8
CentOS Stream	8
Red Hat Enterprise Linux (RHEL)	7, 8 and 9
SUSE Linux Enterprise Server (SUSE)	12 and 15
Ubuntu	18.04 LTS, 20.04 LTS, 22.04 LTS, and 24.04 LTS
Windows Server	2012 R2, 2016, 2019, and 2022



EC2 Image Builder Pipeline Elements

Image Recipe



Image Recipe Properties
Name
Version
OS Type
Components
Storage

Infrastructure Configuration



Configuration Characteristics
Name
Instance Type
SNS
Logs - S3

EventBridge Bus



EventBridge Elements
Event Bus Name
Rules

Distribution Settings



Distribution Settings
Regions
Target Accounts

Work Flows



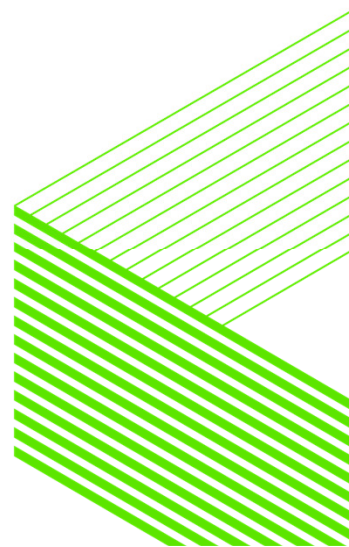
Default Work Flow Steps
Launch Build Instance
Apply Build Components
Inventory Collection
Run Sanitize Script
Run Sys Prep Script
Create Output AMI
Terminate Build Instance

Pipeline Characteristics
Description
IAM Role
Build Schedule

Pipeline

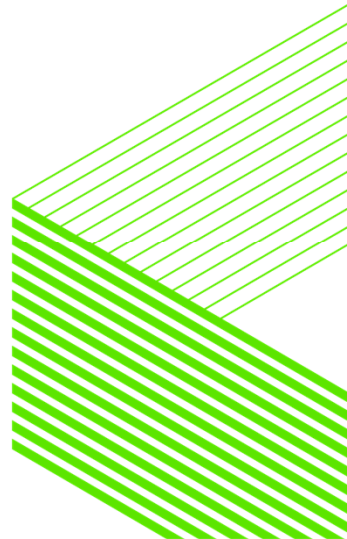


Output Image



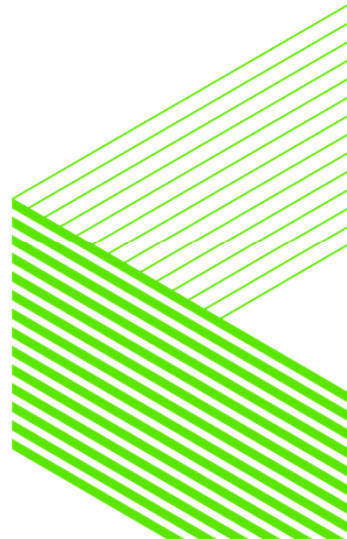
EC2 Image Builder Pipeline Elements

- An image pipeline consists of the following components:
 - Image Recipe
 - Version
 - OS Type – Linux or Windows
 - Components
 - Add Amazon-managed components – aws-cli-version-2-linux in our example
 - This is where all the magic happens – where you write/include your scripts
 - Disk storage
 - Infrastructure Configuration
 - Select Instance Type used for Building the Output AMI
 - Simple Notification Service
 - S3 Location for the builder logs



EC2 Image Builder Pipeline Elements (cont.)

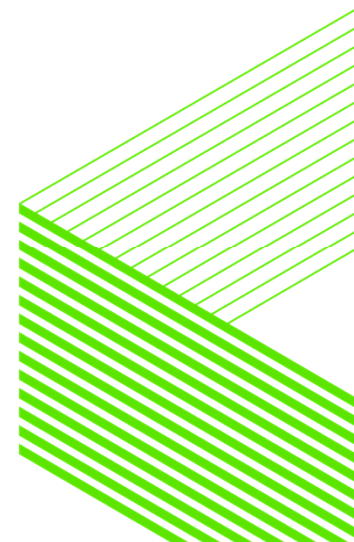
- Event Bridge
 - Event Bus Name – Default is what I only used
 - Filter rules associated with the Event Bus
- Distribution Settings
 - Regions – Specify which region(s) you want to distribute the AMI or container
 - Specify encryption key, target accounts, launch template configuration



EC2 Image Builder Pipeline Elements (cont.)

■ Work Flows

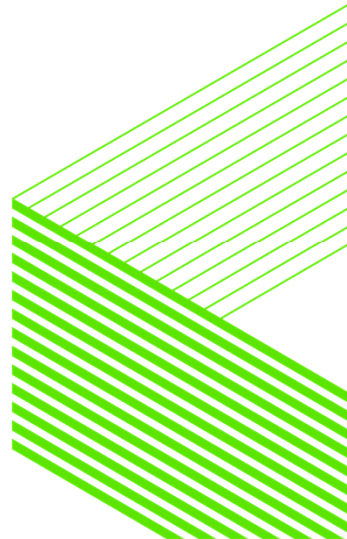
- Build Work Flows – default template steps
 - Launch Build Instance
 - Apply Build Components – This is where disaster happens when your scripts fail or have bugs
 - Inventory Collection – Collect Image Metadata
 - Run Sanitize Script – Sanitizes the instance
 - Run System Prep Script – prepares the instance for a system
 - Create Output AMI – creates the output AMI
 - Terminate Build Instance



EC2 Image Builder Pipeline Elements Work Flows (cont.)

■ Work Flows

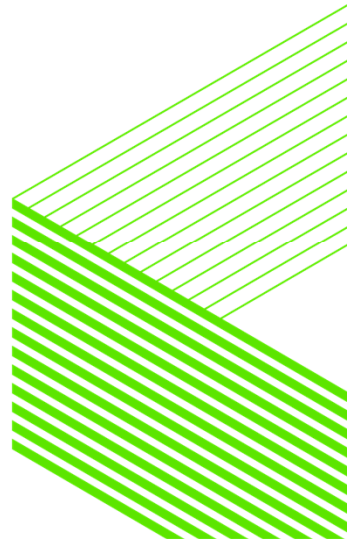
- Test Work Flows - default template steps
 - Launch Test Instance
 - Collect Image Scan Findings
 - Apply Test Components – this is where the magic happens hopefully 😊
 - Terminate Test Instance



Create Custom Component Demonstration

- Objective – Create an AMI with OpenEdge 12.2 or 12.8 on it
- Work Flow consists of a:
 - Build Image
 - Test Image
- Installation Files and Scripts will be stored on S3

File Type	OE 12.2	OE 12.8
Java Files	JDK11x64LNX.tar	JDK17x64.tar.gz
OpenEdge Files	OE122LNX64.tar.gz	OE128LNX64.tar.gz
Linux Scripts	s3cp122-1.sh s3cp122-2.sh	s3cp128-1.sh s3cp128-2.sh
License Batch Scripts	response122.ini	response128.ini



Create Custom Component Demonstration (cont.)

- YAML is used to provide instruction in a component
- One component will be used to produce either a OE 12.2 or a 12.8 version
- This is done through a parameter called version

```
name: Install OpenEdge 12
```

```
description: Installs either 12.2 or 12.8
```

```
schemaVersion: 1.0
```

```
parameters:
```

```
  - version:
```

```
    type: string
```

```
    default: '122'
```

```
    description: The OpenEdge Version 122 is OE 12.2.
```

- During component execution, the parameter version is substituted phases section below:

```
{{version}}
```



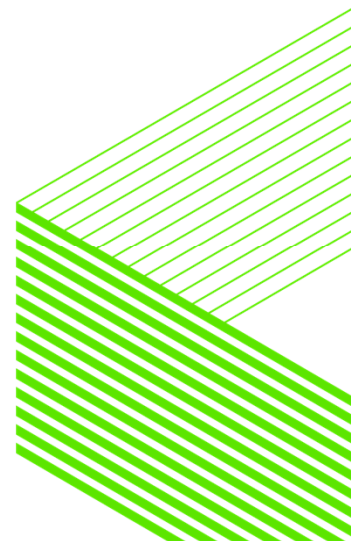
Create Custom Component Demonstration (cont.)

- Define the steps needed within the component's build phase

```
- name: InitialDownload
  action: S3Download
  inputs:
    - source: s3://ec2imagebuilder-pga/s3cp{{version}}-1.sh
      destination: /tmp/s3cp{{version}}-1.sh
- name: FirstInstall
  action: ExecuteBash
  inputs:
    commands:
      - sudo chmod +x /tmp/s3cp{{version}}-1.sh
      - sudo /tmp/s3cp{{version}}-1.sh
```

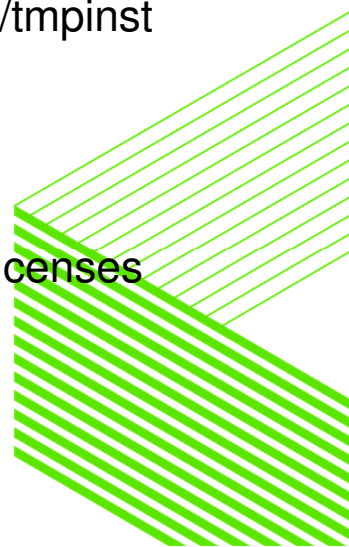
- S3Download and ExecuteBash are two action modules that may be used

<https://docs.aws.amazon.com/imagebuilder/latest/userguide/toe-action-modules.html>



Create Custom Component Demonstration (cont.)

- Two inputs are required for the action S3Download, the source and the target
- The Image recipe default working directory of /tmp is used
- Anything put in /tmp is removed after the process is complete
- After the InitialDownload step, the FirstInstall step is executed with an ExecuteBash action
- The chmod command gives execution rights to the s3cp122-1.sh or s3cp128-1.sh script.
- In s3cp128-1.sh, 3 directories are created: /tmp2/dlc/inst
- This allows the MainDownload step to download the necessary files into /tmp2/dlc/tmpinst directory structure.
- Finally, in the SecondInstall step, both Java (11 or 17) and Progress (12.2 or 12.8) respectively are installed.
- The response122.ini or response128.ini is used for batch load of the OpenEdge Licenses



Create Custom Component Demonstration (cont.)

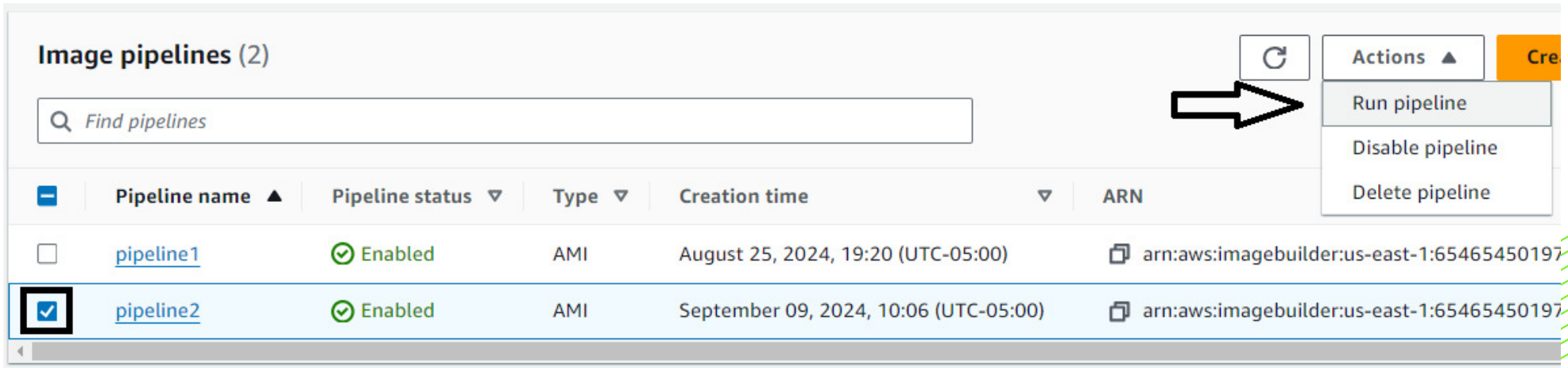
- Next, the test phase is executed
- The s3cptest.sh script is downloaded and is the same for both 12.2 and 12.8
- This script downloads a procedure dispcust.p:

```
output to dispcust.txt.  
for each customer:  
  display custnum name creditlimit.  
end.
```

- It then creates the sports2020 database in the working directory, and runs a dispcust.p as a batch procedure, which creates the output report dispcust.txt.
- This is then uploaded to S3 for verification that OpenEdge was installed correctly.

Running the pipeline

- From the console:



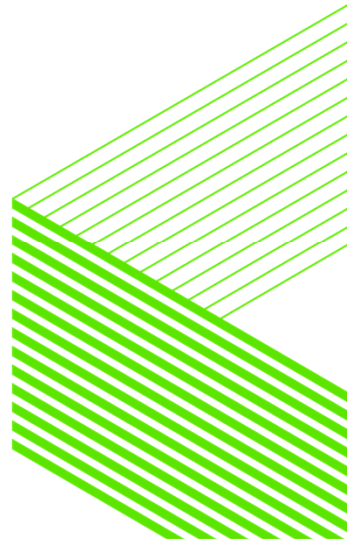
The screenshot shows the AWS Image Builder console interface. At the top, there is a header "Image pipelines (2)" and a search bar labeled "Find pipelines". Below the search bar is a table with columns: Pipeline name, Pipeline status, Type, Creation time, and ARN. Two pipelines are listed: "pipeline1" and "pipeline2". The "pipeline2" row is selected, indicated by a blue highlight and a checked checkbox in the first column. To the right of the table, there is a "Run pipeline" button with a refresh icon. A large black arrow points from this button to a context menu that is open, showing options: "Run pipeline", "Disable pipeline", and "Delete pipeline".

<input type="checkbox"/>	Pipeline name ▲	Pipeline status ▼	Type ▼	Creation time ▼	ARN
<input type="checkbox"/>	pipeline1	✔ Enabled	AMI	August 25, 2024, 19:20 (UTC-05:00)	arn:aws:imagebuilder:us-east-1:65465450197
<input checked="" type="checkbox"/>	pipeline2	✔ Enabled	AMI	September 09, 2024, 10:06 (UTC-05:00)	arn:aws:imagebuilder:us-east-1:65465450197

Running the pipeline

- From the command line:

```
aws imagebuilder start-image-pipeline-execution ^  
  --image-pipeline-arn arn:aws:imagebuilder:us-east-1:654654501973:image-pipeline/pipeline2
```



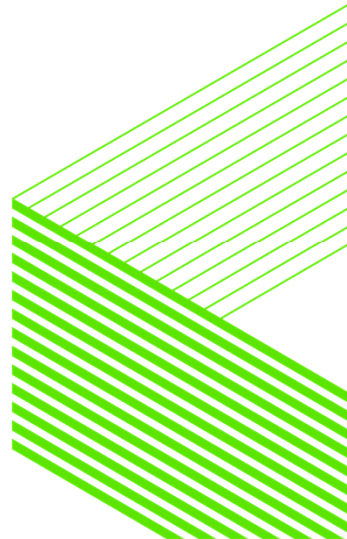
Running the pipeline

- Upon successful pipeline execution, an AMI will be created.

Amazon Machine Images (AMIs) (2) [Info](#) ↻ 🗑️ Recycle Bin 🔗 EC2 Instance Profile

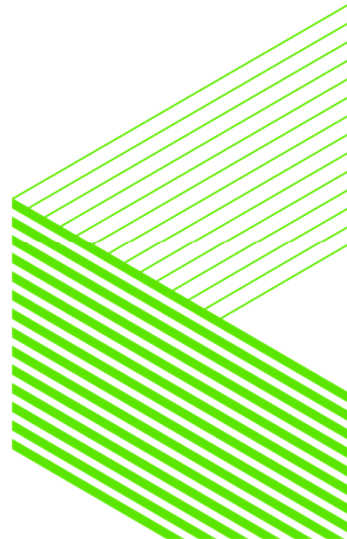
Owned by me ▼

<input type="checkbox"/>	Name ✎	AMI name	AMI ID
<input type="checkbox"/>	OE12.2	ubuntu 2024-09-23T21-50-31.647302Z	ami-0dd75545b8b4f524f
<input type="checkbox"/>	OE12.8	ubuntu 2024-09-23T20-45-13.259584Z	ami-0b2cc8aebbbb91843



Troubleshooting the Pipeline

- AWS provides many logs for identifying issues with the pipeline process
- Here are 3 major areas
 - Cloud Watch Logs
 - Infrastructure Configuration Logs – S3
 - Simple Notification Service



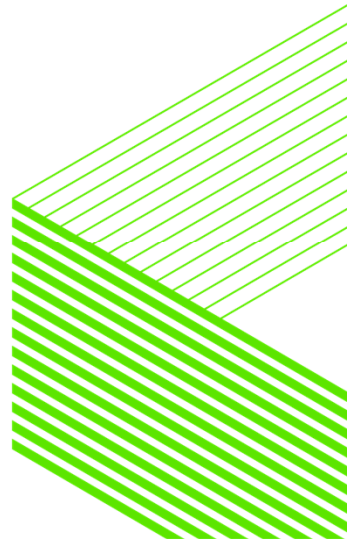
Troubleshooting the Pipeline

- Cloud Watch Logs

- To access, go to CloudWatch->/aws/imagebuilder/<recipename> -> select Log stream

```
▼ 2024-09-25T01:07:18.533Z S3Download: Source:s3://ec2imagebuilder-pga/s3cp122-1.sh,Destination:/tmp/s3cp122-1.sh
```

```
S3Download: Source:s3://ec2imagebuilder-pga/s3cp122-1.sh,Destination:/tmp/s3cp122-1.sh
```



Troubleshooting the Pipeline

■ Infrastructure Configuration Logs – S3

- Application Log

2024-09-09T15:52:35.497387625Z Debug Executor: STARTED STEP InitialDownload in PHASE build

2024-09-09T15:52:35.497416355Z Step InitialDownload

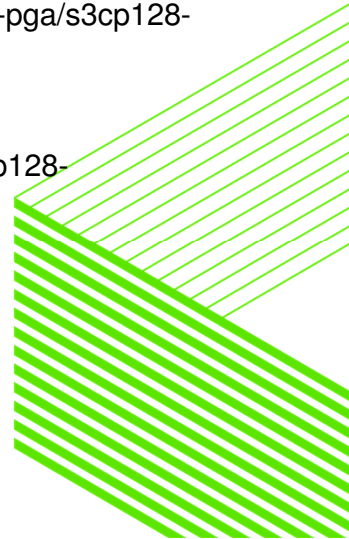
2024-09-09T15:52:35.497429070Z Debug Executor: Starting point of step execution: Step – InitialDownload

2024-09-09T15:52:35.497437581Z Debug Executor: Before resolving parameter inputs [{"source":"s3://ec2imagebuilder-pga/s3cp128-1.sh","destination":"/tmp","expectedBucketOwner":"","overwrite":null}] of step InitialDownload.

2024-09-09T15:52:35.497449406Z Debug Executor: Before unchaining inputs [{"source":"s3://ec2imagebuilder-pga/s3cp128-1.sh","destination":"/tmp","expectedBucketOwner":"","overwrite":null}] of step InitialDownload...

2024-09-09T15:52:35.497459946Z Debug Chaining: STARTED PROCESS FOR UNCHAINING

2024-09-09T15:52:35.497467383Z Debug Chaining: FINISHED PROCESS FOR UNCHAINING



Troubleshooting the Pipeline

- Infrastructure Configuration Logs – S3
 - Console Log – show the output of executed commands

2024-09-23T22:09:37.020409470Z Step ExecuteTest

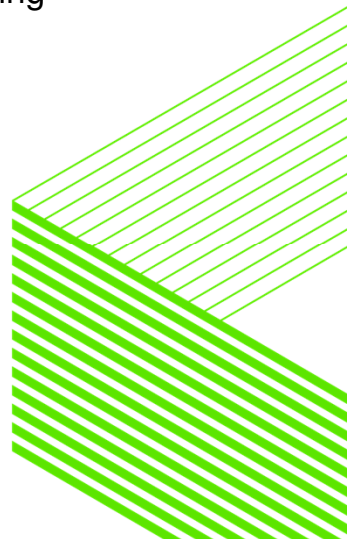
2024-09-23T22:09:37.020811938Z ExecuteBash: STARTED EXECUTION

2024-09-23T22:09:50.262134880Z Stdout: Completed 84 Bytes/84 Bytes (2.3 KiB/s) with 1 file(s) remaining
download: s3://ec2imagebuilder-pga/dispcust.p to ./dispcust.p

2024-09-23T22:09:53.334038512Z Stdout: Procopy session begin for root on batch. (451)

2024-09-23T22:09:55.362599869Z Stdout: Database copied from /usr/dlc122/sports2020. (1365)

2024-09-23T22:09:55.378286184Z Stdout: Procopy session end. (334)



Troubleshooting the Pipeline

- Infrastructure Configuration Logs – S3
 - Chaining JSON file

```
"steps": [  
  {  
    "name": "InitialDownload",  
    "ifConditionResult": "",  
    "inputs": "[{\\"source\\":\\"s3://ec2imagebuilder-pga/s3cp{{version}}-  
1.sh\\",\\"destination\\":\\"/tmp/s3cp{{version}}-  
1.sh\\",\\"expectedBucketOwner\\":\\"\\",\\"overwrite\\":null}]",  
    "outputs": "[null]",  
    "loop": null,  
    "documents": null  
  },  
  {  
    "name": "FirstInstall",  
    "ifConditionResult": "",  
    "inputs": "[{\\"commands\\":[\\"sudo chmod +x /tmp/s3cp{{version}}-1.sh\\",\\"sudo /tmp/s3cp{{version}}-  
1.sh\\"]}]",  
    "outputs": "[{\\"stdout\\":\\"\\"}]",  
    "loop": null,  
    "documents": null  
  },  
],
```

Troubleshooting the Pipeline

- Infrastructure Configuration Logs – S3
 - Component YAML File

phases:

- name: build

steps:

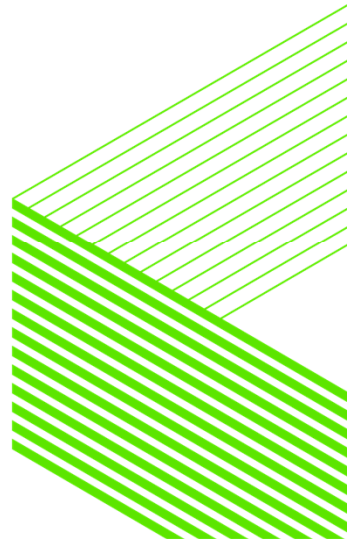
- name: InitialDownload

action: S3Download

inputs:

- source: s3://ec2imagebuilder-pga/s3cp{{version}}-1.sh

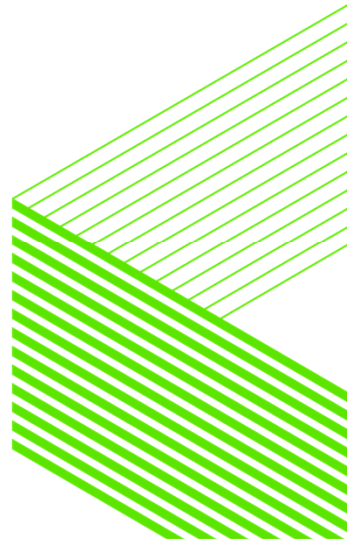
destination: /tmp/s3cp{{version}}-1.sh



Troubleshooting the Pipeline

- Infrastructure Configuration Logs – S3
 - Detailedoutput.json

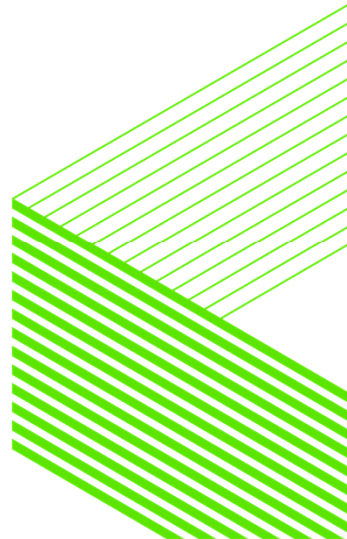
```
{  
  "name": "Install OpenEdge 12",  
  "filePath": "arn:aws:imagebuilder:us-east-1:654654501973:component/ubuntucomponent/1.1.23/1",  
  "status": "success",  
  "description": "Installs either 12.2 or 12.8",  
  "startTime": "2024-09-23T22:09:36Z",  
  "endTime": "2024-09-23T22:10:00Z",  
  "disableLogging": false,  
  "failureMessage": "",  
}
```



Troubleshooting the Pipeline

- Simple Notification Service – JSON file

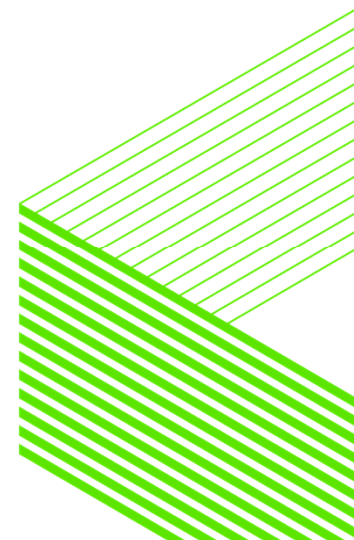
```
"sourcePipelineArn": "arn:aws:imagebuilder:us-east-1:654654501973:image-pipeline/pipeline2",  
  "infrastructureConfiguration": {  
    "logging": {  
      "s3Logs": {}  
    },  
    "keyPair": "pga25",  
    "instanceProfileName": "imagebuilder",  
    "accountId": "654654501973",  
    "dateUpdated": "Jun 23, 2024 6:15:43 PM",  
    "terminateInstanceOnFailure": false,  
    "dateCreated": "Jun 22, 2024 5:35:56 PM",  
    "subnetId": "subnet-083f8fa74b0f2ddd5",  
    "securityGroupIds": [  
      "sg-00c9c464c6e07e1a5"  
    ],  
  },  
}
```



EC2 Imagebuilder Costs

- EC2 Image Builder is free to use, but there are costs associated with the underlying AWS resources used to create, store, and share the images.
- By using t2.micro for build and test means that if it's a new account, then the free-tier pricing is available. This means the first 750 hours / month for t2.micro is free.
- On-demand pricing

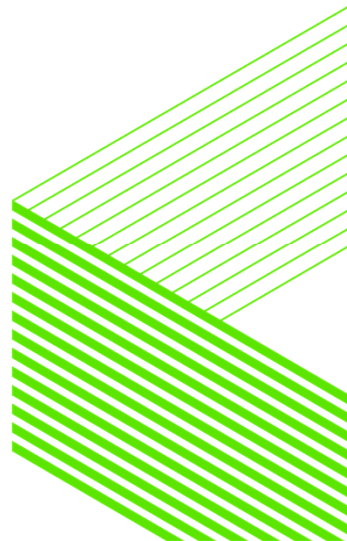
Instance Type	vCPUs	Memory (GiB)	Hourly Pricing
t2.micro	1	1 GiB	\$0.0116/hr
t2.small	1	2 GiB	\$0.023/hr
t2.medium	2	4 GiB	\$0.0464/hr
t2.large	2	8 GiB	\$0.0928/hr



EC2 Imagebuilder Costs (cont.)

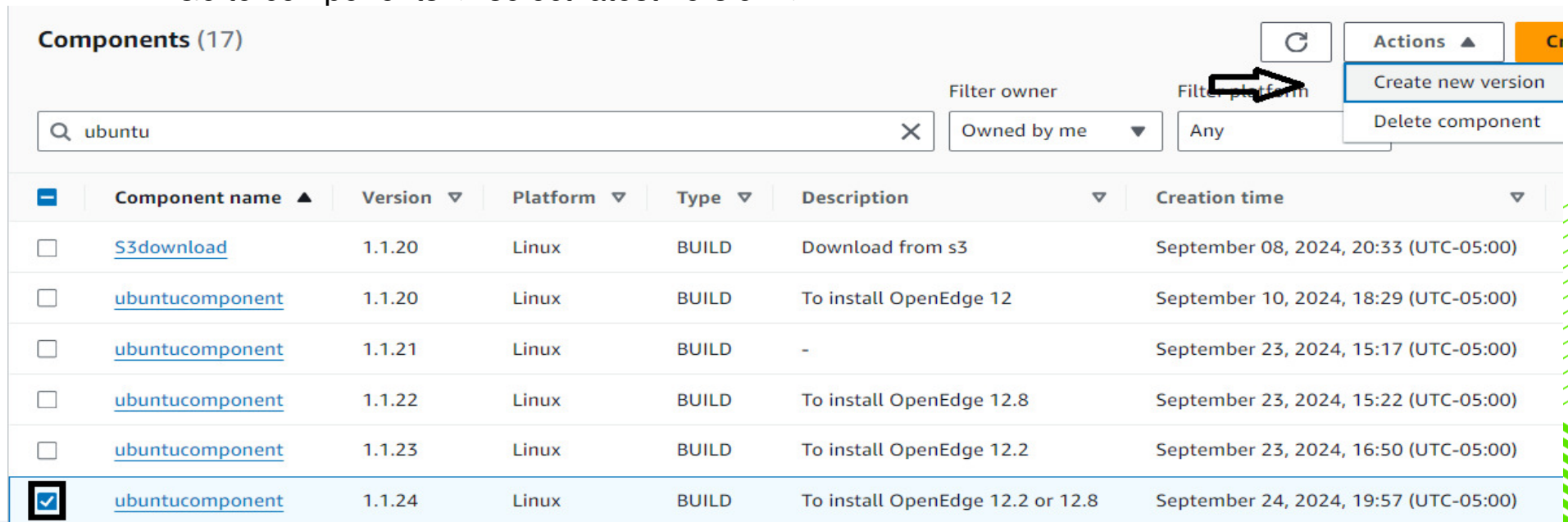
- For storage costs, it is much cheaper to store an AMI than an ec2 instance.
 - Since Imagebuilder generates an AMI, that is the optimal cost until an ec2 instance is launched from the AMI.

Storage	Cost
Ec2 instance SSD (gp3) 100gb	\$.08/gb/month or \$8.00/month
AMI 100gb	\$.05/gb/month or \$5.00/month



Changing Component Parameters

- In our example, the default version is 122.
- How do we change the version to 128?
 - Manually in the console
 - Go to components -> select latest version ->



Components (17)

Search: Filter owner: Owned by me Filter platform: Any

<input type="checkbox"/>	Component name ▲	Version ▼	Platform ▼	Type ▼	Description ▼	Creation time ▼
<input type="checkbox"/>	S3download	1.1.20	Linux	BUILD	Download from s3	September 08, 2024, 20:33 (UTC-05:00)
<input type="checkbox"/>	ubuntucomponent	1.1.20	Linux	BUILD	To install OpenEdge 12	September 10, 2024, 18:29 (UTC-05:00)
<input type="checkbox"/>	ubuntucomponent	1.1.21	Linux	BUILD	-	September 23, 2024, 15:17 (UTC-05:00)
<input type="checkbox"/>	ubuntucomponent	1.1.22	Linux	BUILD	To install OpenEdge 12.8	September 23, 2024, 15:22 (UTC-05:00)
<input type="checkbox"/>	ubuntucomponent	1.1.23	Linux	BUILD	To install OpenEdge 12.2	September 23, 2024, 16:50 (UTC-05:00)
<input checked="" type="checkbox"/>	ubuntucomponent	1.1.24	Linux	BUILD	To install OpenEdge 12.2 or 12.8	September 24, 2024, 19:57 (UTC-05:00)

Changing Component Parameters

- How do we change the version to 128?
 - Manually in the console
 - Update version number – in this case to 1.1.25

parameters:

- version:

type: string

default: '128'

description: The OpenEdge Version 128 is OE 12.8.

- The latest image recipe will automatically call the latest version of that component:

[ubuntucomponent](#)
View parameter
values

1.1.25

To install OpenEdge 12.2
or 12.8

September 25, 2024,
14:52 (UTC-05:00)

Linux

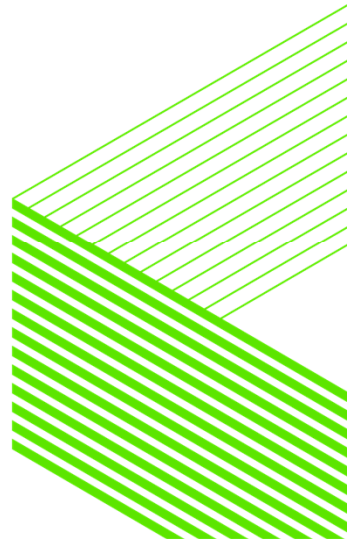


Changing Component Parameters

- How do we change the version to 128 from the command line?
- The windows bat file: createcomponent.bat:

```
aws imagebuilder create-component ^  
  --name ubuntucomponent ^  
  --semantic-version %1 ^  
  --description "To install OpenEdge %2" ^  
  --platform Linux ^  
  --uri s3://ec2imagebuilder-pga/compinstall%2.yaml
```

From the command line: createcomponent 1.1.26 12.8

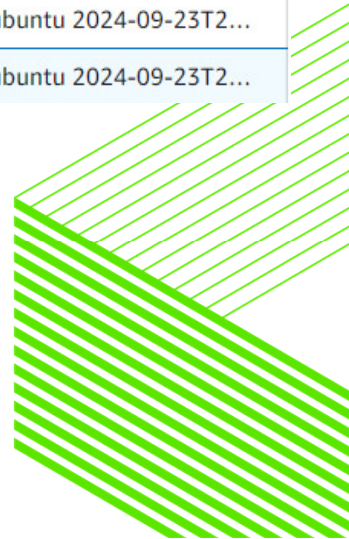


Launching an EC2 instance from an AMI

Amazon Machine Images (AMIs) (1/2) [Info](#) Refresh Recycle Bin EC2 Image Builder Actions Launch instance from AMI

Owned by me < 1 > Settings

Name	AMI name	AMI ID	Source
<input type="checkbox"/> OE12.2	ubuntu 2024-09-23T21-50-31.647302Z	ami-0dd75545b8b4f524f	654654501973/ubuntu 2024-09-23T2...
<input checked="" type="checkbox"/> OE12.8	ubuntu 2024-09-23T20-45-13.259584Z	ami-0b2cc8aebbb91843	654654501973/ubuntu 2024-09-23T2...



Launching an EC2 instance from an AMI

Name and tags [Info](#)

Name
 [A](#)


▼ **Application and OS Images (Amazon Machine Image)** [Info](#)

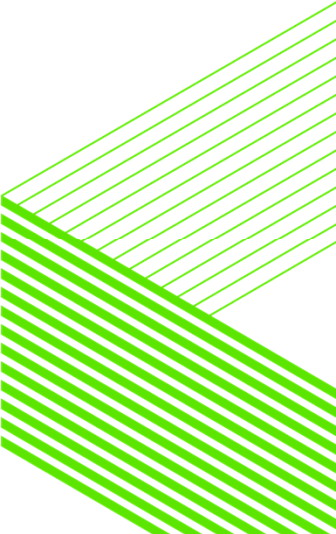
An AMI is a template that contains the software configuration (operating system, application applications) required to launch your instance. Search or Browse for AMIs if you don't see w below

[AMI from catalog](#) | [Recents](#) | [My AMIs](#) | [Quick Start](#)

Name
ubuntu 2024-09-23T20-45-13.259584Z

Description

Image ID
ami-0b2cc8aebbbb91843 



Launching an EC2 instance from an AMI

Key pair name - *required*

ubuntu

▼ Network settings [Info](#)

VPC - *required* [Info](#)

vpc-0c27b644c3dffa7d8 (VPCA)
10.0.0.0/16

Subnet [Info](#)

subnet-083f8fa74b0f2ddd5 VPCASUBPUBLIC1A
VPC: vpc-0c27b644c3dffa7d8 Owner: 654654501973 Availability Zone: us-east-1a
Zone type: Availability Zone IP addresses available: 250 CIDR: 10.0.0.0/24

Auto-assign public IP [Info](#)

Enable

Additional charges apply when outside of **free tier allowance**

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to your instance.

Create security group Select existing security group

Common security groups [Info](#)

Select security groups

SSH sg-00c9c464c6e07e1a5 X
VPC: vpc-0c27b644c3dffa7d8

Security groups that you add or remove here will be added to or removed from all your network interfaces.



Create a template from an instance

The screenshot shows the AWS Management Console interface for instances. At the top, there's a header 'Instances (1/1) Info' with a refresh button and 'Last updated less than a minute ago'. Below this is a search bar and a filter dropdown set to 'All states'. A filter 'Instance state = running' is applied. The main table lists one instance: 'ubuntu-oe128' with ID 'i-0515b988a5685eccb', state 'Running', and type 't2.micro'. A context menu is open over the instance, with an arrow pointing to the 'Create template from instance' option. The menu also includes 'Create image' and 'Launch more like this'. On the right, the 'Actions' dropdown is open, showing options like 'Connect', 'View details', 'Manage instance state', 'Instance settings', 'Networking', 'Security', 'Image and templates', and 'Monitor and troubleshoot'.

Name	Instance ID	Instance state	Instance type
ubuntu-oe128	i-0515b988a5685eccb	Running	t2.micro

Create a template from an instance

[EC2](#) > [Launch templates](#) > Create template from instance

Create launch template

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

Launch template name and description

Source instance

i-0515b988a5685eccb

Launch template name - *required*

MyOE12.8-Template



Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.

Template version description

A prod webserver for MyApp

Max 255 chars

Auto Scaling guidance [Info](#)

Select this if you intend to use this template with EC2 Auto Scaling

Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

► **Template tags**

▼ Summary

Software Image (AMI)

ubuntu 2024-09-23T20-45-13.259...[read more](#)
ami-0b2cc8aeb91843

Virtual server type (instance type)

t2.micro

Firewall (security group)

SSH

Storage (volumes)

1 volume(s) - 10 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel

Create launch template

Launch template contents

Launch an instance from a template

- Select Launch Templates -> select desired template -> Actions -> Launch instance from template

EC2 Dashboard X

EC2 Global View

Events

Console-to-Code [Preview](#)

▼ Instances

Instances

Instance Types

Launch Templates

Spot Requests

Launch Templates (1/2) Info

Search

	Launch Template ID	Launch Template Name	Default Version	Latest Version
<input type="checkbox"/>	lt-09bb36e873735c542	oe128template	1	1
<input checked="" type="checkbox"/>	lt-0de6595b5242636da	MyOE12.8-Template	1	1

Actions ▲ Create launch template

- Launch instance from template
- Modify template (Create new version)
- Delete template
- Delete template version
- Set default version
- Manage tags
- Create Spot Fleet
- Create Auto Scaling group
- View details

Launch instance from template (cont.)

EC2 > Launch templates > Launch instance from template

Launch instance from template

Launching from a template allows you to launch from an instance configuration that you would have saved in the past. These saved configurations can be reused and shared with other users to standardize launches across an organisation.

Choose a launch template

Source template

MyOE12.8-Template
ID: lt-0de6595b5242636da

1 (Default)

Instance details

Your instance details are listed below. Any fields that are not specified as part of the configuration below will use the template or default values for those fields. Ensure that you have permissions to override these parameters or your instance launch will fail.

Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Summary

Number of instances [Info](#)

1


ubuntu 2024-09-23T20-45-13.259...[read more](#)
ami-0b2cc8aeb91843

Virtual server type (instance type)
t2.micro

Firewall (security group)
SSH

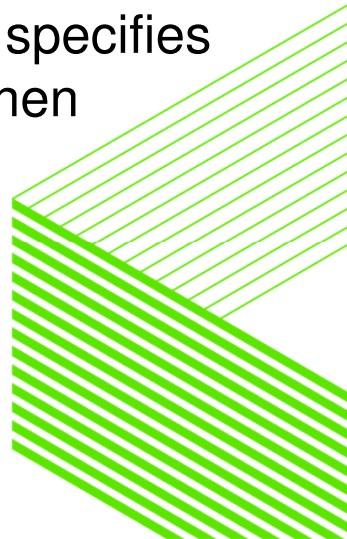
Storage (volumes)
1 volume(s) - 10 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB

Cancel  **Launch instance**

Using CloudFormation to create instances

- Once the ec2 imagebuilder pipeline finishes running and creates the AMI, CloudFormation may be also used to create instances from these AMIs.
- What is CloudFormation?
 - It's a service that helps model and setup AWS resources using templates.
 - These templates describe all the AWS resources necessary for your cloud environment.
 - CloudFormation handles the provisioning and configuring of those resources for you.
- Creating a stack involves deploying a CloudFormation template that specifies the cloud resources and their configuration, which CloudFormation then provisions and configures.



Simple ec2 template - cf-ec2-ubuntu-oe128-1a.yaml

- This creates an ec2 instance for OE12.8 in the VPCASUBPUBLIC1A subnet using the SSH security group

Resources:

EC2Instance:

Type: AWS::EC2::Instance

Properties:

IamInstanceProfile: Session-Mgr-Role

AvailabilityZone: us-east-1a

ImageId: ami-0b2cc8aeb91843 <- 12.8 AMI

KeyName: ubuntu

InstanceType: t2.micro

NetworkInterfaces:

- AssociatePublicIpAddress: true

DeviceIndex: 0

SubnetId: subnet-083f8fa74b0f2ddd5 <- VPCASUBPUBLIC1A - Public Subnet 1A

GroupSet:

- sg-00c9c464c6e07e1a5 <- SSH access security access

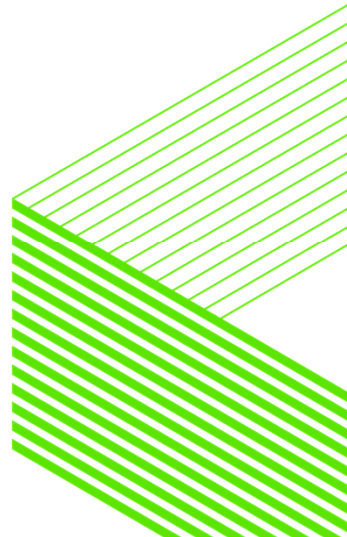
DeleteOnTermination: true

Tags:

- Key: Name

Value: ubuntu-oe128-1a-cloudformation

(name of ec2 instance as it appears in the AWS console after it's created.)



Creating a stack from a template

- In the CloudFormation service, select the button “Create stack”.
- Select Upload a template file and select the “Choose file” button and select the desired yaml file, then click next.

Create stack

Prerequisite - Prepare template
You can also create a template by scanning your existing resources in the [IaC generator](#).

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Choose an existing template
Upload or choose an existing template.

Use a sample template
Choose from our sample template library.

Build from Application Composer
Create a template using a visual builder.

Specify template [Info](#)
A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL
Provide an Amazon S3 URL to your template.

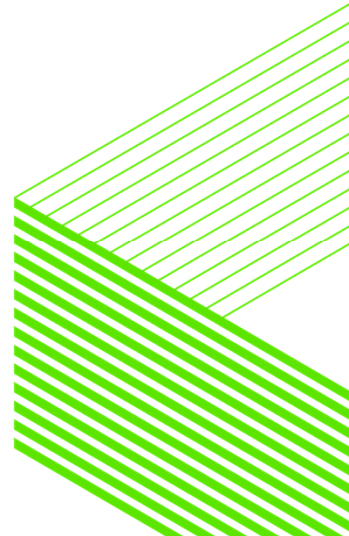
Upload a template file
Upload your template directly to the console.

Sync from Git
Sync a template from your Git repository.

Upload a template file

cf-ec2-ubuntu-oe128-1a.yaml

JSON or YAML formatted file



Creating a stack

Enter a unique stack name and select next, click next again and then select Submit.

Specify stack details

Provide a stack name

Stack name

ec2-stack-1



Stack name must be 1 to 128 characters, start with a letter, and only contain alphanumeric characters. Character count: 11/128.

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

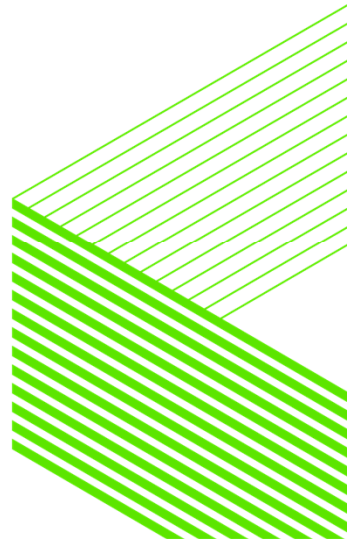
No parameters

There are no parameters defined in your template

Cancel

Previous

Next



Creating a stack

- Follow the events to see if the process completes normally.
- Once the status CREATE_COMPLETE appears, then check for any new instance created.

ec2-stack-1 ⚙️ | >

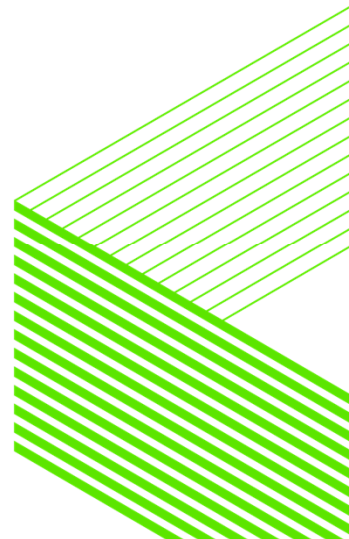
Delete Update Stack actions ▼ Create stack ▼

< Stack info **Events** Resources Outputs Parameters Template Char >

Events (7) Detect root cause ↻

⚙️

Timestamp	Logical ID	Status	Detailed status
2024-09-26 14:55:26 UTC-0500	ec2-stack-1	✔️ CREATE_COMPLETE	-
2024-09-26 14:55:25 UTC-0500	EC2Instance	✔️ CREATE_COMPLETE	-
2024-09-26 14:55:16 UTC-0500	ec2-stack-1	ⓘ CREATE_IN_PROGRESS	✔️ CONFIGURATION COMPLETE
2024-09-26 14:55:16 UTC-0500	EC2Instance	ⓘ CREATE_IN_PROGRESS	✔️ CONFIGURATION COMPLETE
2024-09-26 14:55:14 UTC-0500	EC2Instance	ⓘ CREATE_IN_PROGRESS	-



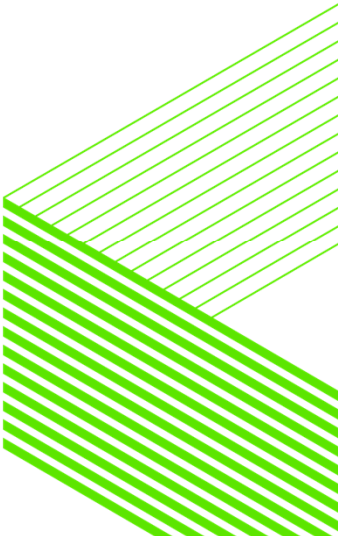
Check for new instance created

Instances (1) [Info](#) Last updated less than a minute ago Refresh Connect Instance state ▼ Actions

All states ▼

Instance state = running X Clear filters

<input type="checkbox"/>	Name ✎	Instance ID	Instance state ▼	Instance type
<input type="checkbox"/>	ubuntu-oe128-1a-cloudformation	i-0c1657eb57837c84b	Running 🔍 🔍	t2.micro

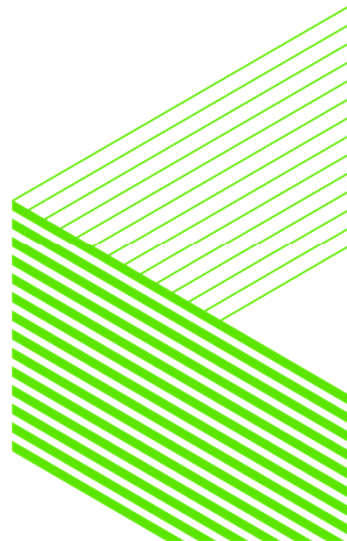


Summary

- After identifying the tasks required to create OpenEdge servers, manually perform these tasks and record every step.
- Use EC2 Image Builder to automate this process to generate an AMI.
- Generate ec2 OpenEdge servers by:
 - Launching instances from Amazon Machine Images
 - Launching instances from Launch Templates
 - Launching instances from a CloudFormation Stack
- To learn more about AWS:

<https://docs.aws.amazon.com/>

- To learn more about ec2 image builder and cloud formation:
 - <https://docs.aws.amazon.com/imagebuilder/>
 - <https://docs.aws.amazon.com/cloudformation/>



Questions

