# H-Containers on AWS Tutorial @ VT

First Popcorn Summit
8th November, 2019

VIRGINIA TECH

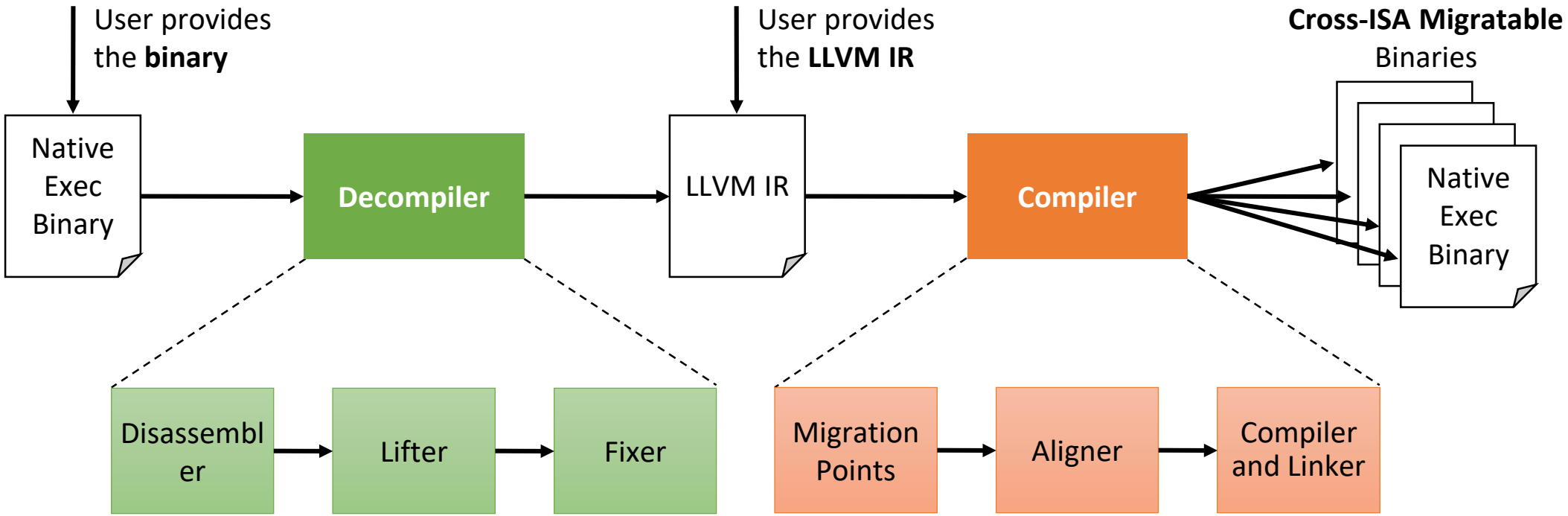
STEVENS INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY

# Popcorn Linux …

- **Usability? Somewhat easy**
  - No modifications required to source-files
  - Modifications required to the compilation scripts
- **Compatibility? Lacks support for …**
  - Language virtual machines (but it principle it should work)
  - Server applications
  - Dynamically linked executable binaries
- **Deployability? Restrictive for large deployments**
  - Requires a specific compiler toolchain
  - Requires a specific Linux kernel version
    - That supports and handful of machines

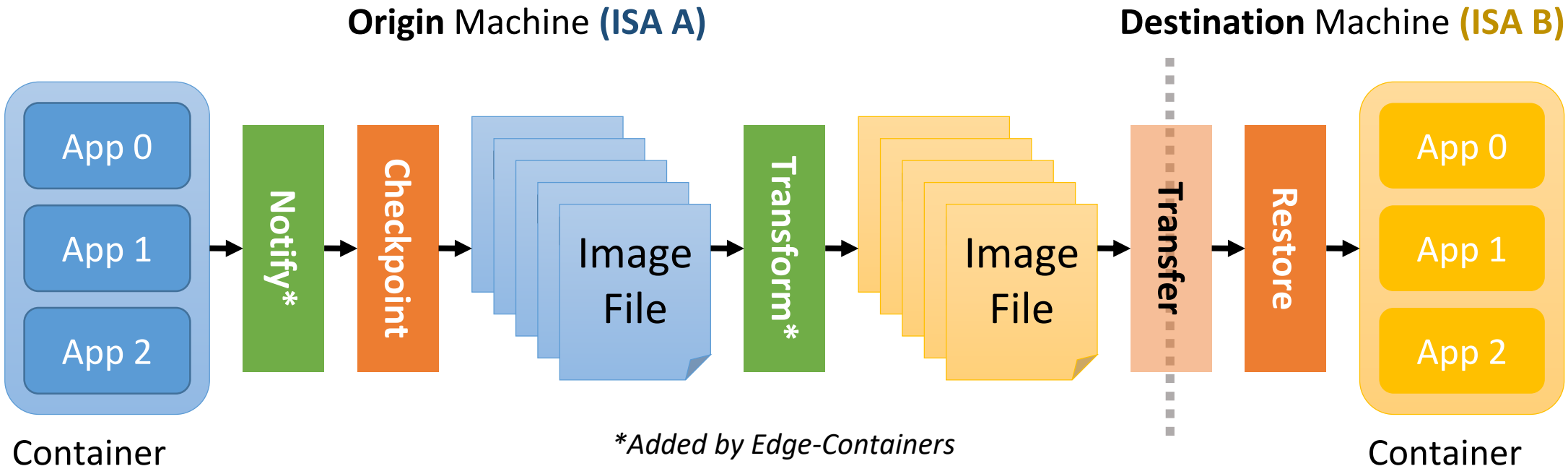
# Welcome H-Containers! #1

- Migrate applications among processors of diverse ISA
- **Without any modification to the application source code**
  - Applications are automatically transformed to run on multiple ISAs



# Welcome H-Containers! #2

- Migrate applications among processors of diverse ISA
- **Without any reliance of a specific Linux version (or kernel patches)**
  - Edge-Containers extends Linux's Checkpoint Restart in User Space (CRIU)



# H-Containers – Summary

- Enables programs migration between heterogeneous ISA CPUs
  - Forth and back
  - On any Linux kernel version that supports CRIU
  - Any binary
    - Currently, FPU is not supported
- Simultaneous thread execution on multiple ISA CPUs requires Popcorn Linux kernel
  - But the automatic binary executable transformation can be exploited
- Will be released open source soon

# Tutorial Overview

- Part1
  - **Environment Setup**
- Part2
  - **Hello World!**
- Part3
  - **Redis, a Real-world Application**

# Tutorial Part 1

## Environment Setup

# Overview

1. AWS Setup
2. H-Container Deployment on AWS
3. Connect to AWS
4. Download and Install Components
5. Docker Configuration
6. CRIU Compilation
7. Check Installation is Working
8. Tutorial Material

# AWS Setup #1


- Login to your AWS account
- Go to the AWS Management Console
- Pick the server location by selecting the drop-down menu
  - Located on the top-left console (see next slide)
  - Note that North California doesn't have ARM AMI
- Click EC2 to enter the EC2 Dashboard (see next-next slide)

# AWS Management Console

## AWS services

**Find Services**
You can enter names, keywords or acronyms.

Example: Relational Database Service, database, RDS

**Recently visited services**

EC2 Billing S3 IAM

**All services**

## Build a solution

Get started with simple wizards and automated workflows.

**Launch a virtual machine**
With EC2
2-3 minutes

**Build a web app**
With Elastic Beanstalk
6 minutes

**Build using virtual servers**
With Lightsail
1-2 minutes

**Connect an IoT device**
With AWS IoT
5 minutes

**Start a development project**
With CodeStar
5 minutes

**Register a domain**
With Route 53
3 minutes

**Deploy a serverless microservice**
With Lambda, API Gateway
2 minutes

**Host a static web app**
With AWS Amplify Console
5 minutes

## Access resources on the go

Access the Management Cor
Console Mobile App. Learn

## Explore AWS

**Free Digital Training**
Get access to 350+ self-paced online
products and services. Learn more

**Amazon RDS**
Set up, operate, and scale your relati
cloud. Learn more

**AWS Security Hub**
Centrally view and manage security alerts and automate compliance checks. Learn more

**Stream Live re:Invent Keynotes and Launches, Dec 2 – 6**
Hear from AWS leaders, and learn about new products.
Sign up

## Have feedback?

Submit feedback to tell us about your experience with the AWS Management Console.

US East (N. Virginia)
US East (Ohio)
**US West (N. California)**
US West (Oregon)
Asia Pacific (Hong Kong)
Asia Pacific (Mumbai)
Asia Pacific (Seoul)
Asia Pacific (Singapore)
Asia Pacific (Sydney)
Asia Pacific (Tokyo)
Canada (Central)
EU (Frankfurt)
EU (Ireland)
EU (London)
EU (Paris)
EU (Stockholm)
Middle East (Bahrain)
South America (São Paulo)

EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
Lifecycle Manager
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
LOAD BALANCING
Load Balancers
Target Groups
AUTO SCALING
Launch Configurations
Auto Scaling Groups

## Resources

You are using the following Amazon EC2 resources in the US West (N. California) region:

- 0 Running Instances
- 0 Dedicated Hosts
- 0 Volumes
- 0 Key Pairs
- 0 Placement Groups
- 0 Elastic IPs
- 0 Snapshots
- 0 Load Balancers
- 1 Security Groups

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

## Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US West (N. California) region

## Migrate a Machine

Use CloudEndure Migration to simplify, expedite, and automate large-scale migrations from physical, virtual, and cloud-based infrastructure to AWS.

Get started with CloudEndure Migration

## Quick ID filter

Enter a resource ID

Create filter

## Service Health

**Service Status:**

US West (N. California):

**Availability Zone Status:**

us-west-1a:
Availability zone is operating normally

us-west-1c:
Availability zone is operating normally

Service Health Dashboard

## Scheduled Events

**US West (N. California):**

No events

## Account Attributes

Supported Platforms
VPC

Default VPC
vpc-9d213dfa

Console experiments
Settings

## Additional Information

Getting Started Guide
Documentation
All EC2 Resources
Forums
Pricing
Contact Us

## AWS Marketplace

Find free software trial products in the AWS Marketplace from the EC2 Launch Wizard. Or try these popular AMIs:

Barracuda CloudGen Firewall for AWS - PAYG

By Barracuda Networks, Inc.
Rating ★★★★☆
Starting from $0.60/hr or from $4,599/yr (12% savings) for software + AWS usage fees
View all Infrastructure Software

Matillion ETL for Amazon Redshift

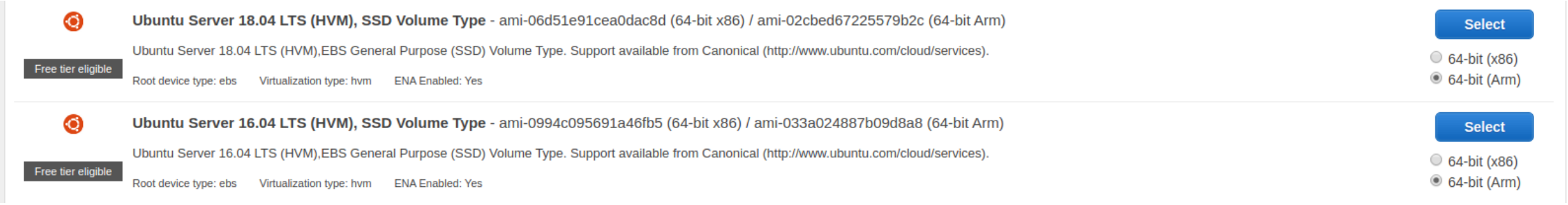
By Matillion
Rating ★★★★★
$1.37 to $5.48/hr for software + AWS usage fees
View all Business Software

Trend Micro Deep Security

By Trend Micro
Rating ★★★★★

# AWS Setup #2

- Click Launch Instance
- Choose default AWS AMI
- Choose Ubuntu 18.04, 64-bit (x86)
  - (16.04 if you want to install popcorn-compiler)
- Repeat for ARM, choose Ubuntu 18.04, 64-bit (Arm)
  - In the tutorial we are using 18.04 (for both ARM and x86)



# Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

**Red Hat Enterprise Linux 8 (HVM), SSD Volume Type** - ami-087c2c50437d0b80d (64-bit x86) / ami-047f9f2f5072dd073 (64-bit Arm)

Red Hat

Free tier eligible

Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

64-bit (Arm)

---

**SUSE Linux Enterprise Server 15 SP1 (HVM), SSD Volume Type** - ami-0e6612c7b620ecf3a (64-bit x86) / ami-04d6f059fbfd089b0 (64-bit Arm)

SUSE Linux

Free tier eligible

SUSE Linux Enterprise Server 15 Service Pack 1 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

64-bit (Arm)

---

**Ubuntu Server 18.04 LTS (HVM), SSD Volume Type** - ami-06d51e91cea0dac8d (64-bit x86) / ami-02cbed67225579b2c (64-bit Arm)

Free tier eligible

Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

64-bit (Arm)

---

**Ubuntu Server 16.04 LTS (HVM), SSD Volume Type** - ami-0994c095691a46fb5 (64-bit x86) / ami-033a024887b09d8a8 (64-bit Arm)

Free tier eligible

Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

64-bit (Arm)

---

**Microsoft Windows Server 2019 Base** - ami-0bff712af642c77c9

Windows

Free tier eligible

Microsoft Windows 2019 Datacenter edition. [English]

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

---

**Deep Learning AMI (Ubuntu 16.04) Version 25.3** - ami-04121e1f9d541d468

MXNet-1.5.0, TensorFlow-1.14, PyTorch-1.2, Keras-2.2, Chainer-6.1, Caffe/2-0.8, Theano-1.0 & CNTK-2.7, configured with NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker & NVIDIA-Docker. For a fully managed experience, check: https://aws.amazon.com/sagemaker

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

---

**Deep Learning AMI (Amazon Linux) Version 25.3** - ami-0b5c8005b478d04e6

Amazon Linux

MXNet-1.5.0, TensorFlow-1.14, PyTorch-1.2, Keras-2.2, Chainer-6.1, Caffe/2-0.8, Theano-1.0 & CNTK-2.7, configured with NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker & NVIDIA-Docker. For a fully managed experience, check: https://aws.amazon.com/sagemaker

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

# AWS Setup #3


- Following steps will ask to configure and select the deployment of each VM
- Based on individual requirements more powerful VMs can be deployed
- For the tutorial
  - Default 1 core, 2GB memory, and 5G disk
- At the end of the configuration, click **Launch**

1. Choose AMI | **2. Choose Instance Type** | 3. Configure Instance | 4. Add Storage | 5. Add Tags | 6. Configure Security Group | 7. Review

# Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

**Filter by:** All instance types | Current generation | Show/Hide Columns

**Currently selected:** a1.medium (- ECUs, 1 vCPUs, 2.3 GHz, AWS Graviton Processor, 2 GiB memory, EBS only)

| | Family | Type | vCPUs | Memory (GiB) | Instance Storage (GB) | EBS-Optimized Available | Network Performance | IPv6 Support |
|---|---|---|---|---|---|---|---|---|
| ☑ | General purpose | a1.medium | 1 | 2 | EBS only | Yes | Up to 10 Gigabit | Yes |
| ☐ | General purpose | a1.large | 2 | 4 | EBS only | Yes | Up to 10 Gigabit | Yes |
| ☐ | General purpose | a1.xlarge | 4 | 8 | EBS only | Yes | Up to 10 Gigabit | Yes |
| ☐ | General purpose | a1.2xlarge | 8 | 16 | EBS only | Yes | Up to 10 Gigabit | Yes |
| ☐ | General purpose | a1.4xlarge | 16 | 32 | EBS only | Yes | Up to 10 Gigabit | Yes |
| ☐ | General purpose | a1.metal | 16 | 32 | EBS only | Yes | Up to 10 Gigabit | Yes |
| ⊘ | General purpose | t2.nano | 1 | 0.5 | EBS only | - | Low to Moderate | Yes |
| ⊘ | General purpose | t2.micro Free tier eligible | 1 | 1 | EBS only | - | Low to Moderate | Yes |
| ⊘ | General purpose | t2.small | 1 | 2 | EBS only | - | Low to Moderate | Yes |
| ⊘ | General purpose | t2.medium | 2 | 4 | EBS only | - | Low to Moderate | Yes |
| ⊘ | General purpose | t2.large | 2 | 8 | EBS only | - | Low to Moderate | Yes |
| ⊘ | General purpose | t2.xlarge | 4 | 16 | EBS only | - | Moderate | Yes |
| ⊘ | General purpose | t2.2xlarge | 8 | 32 | EBS only | - | Moderate | Yes |

# Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

**Improve your instances' security. Your security group, launch-wizard-2, is open to the world.**

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.

You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

## AMI Details

Edit AMI

**ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-arm64-server-20190627.1 - ami-0606a0d9f566249d3**

Canonical, Ubuntu, 18.04 LTS, arm64 bionic image build on 2019-06-27

Root Device Type: ebs Virtualization type: hvm

## Instance Type

Edit instance type

| Instance Type | ECUs | vCPUs | Memory (GiB) | Instance Storage (GB) | EBS-Optimized Available | Network Performance |
|---|---|---|---|---|---|---|
| a1.medium | - | 1 | 2 | EBS only | Yes | Up to 10 Gigabit |

## Security Groups

Edit security groups

| Security Group ID | Name | Description |
|---|---|---|
| sg-0ab23742f1cdf5487 | launch-wizard-2 | launch-wizard-2 created 2019-07-18T10:30:51.374-04:00 |

**All selected security groups inbound rules**

| Type | Protocol | Port Range | Source | Description |
|---|---|---|---|---|
| All traffic | All | All | 0.0.0.0/0 | |
| All traffic | All | All | ::/0 | |
| SSH | TCP | 22 | 0.0.0.0/0 | |
| SSH | TCP | 22 | 155.246.151.34/32 | |

## Instance Details

Edit instance details

**Number of instances** 1

**Network** vpc-6ee70605

**Subnet** subnet-47aeeb2d

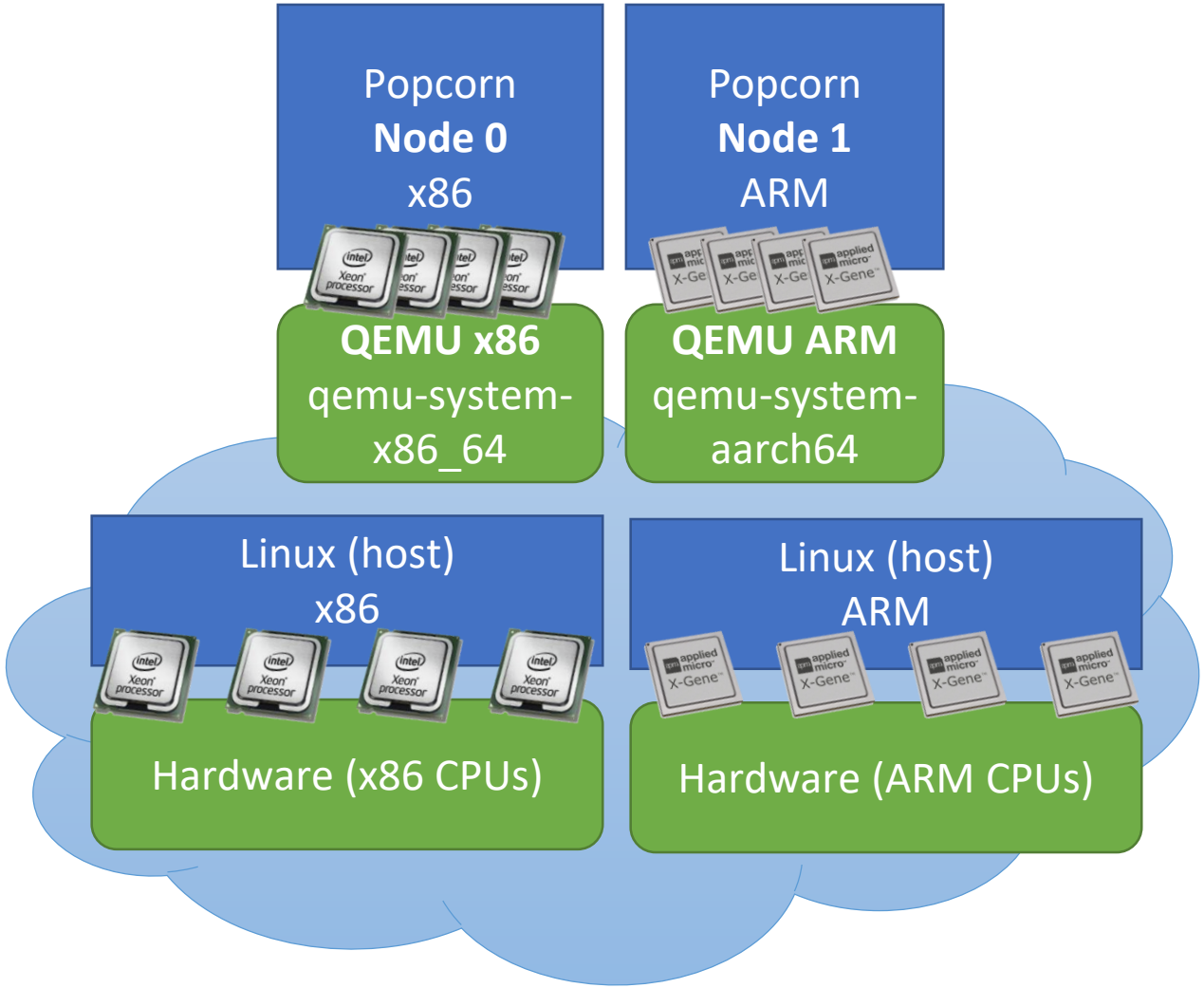
**Purchasing option** On demand

Cancel Previous Launch

# H-Container Deployment on AWS

- 2x AWS cloud EC2 VM instance
- x86 Instance
  - AWS **Ubuntu Server 18.04 LTS x86**
- ARM Instance
  - AWS **Ubuntu Server 18.04 LTS ARM**

- Interact with instances via SSH



# Connect to AWS #1

- Refer to AWS instructions https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstances.html

- To connect via SSH

```
$ ssh -i “$your key file” ec2-user@public_ip
```

- Example

```
$ ssh -i “docker.pem” ubuntu@18.223.151.11
```

***.pem and IP provided now!**

# Connect to AWS #2

Desktop — ubuntu@ip-172-31-44-28: ~ — ssh -i docker.pem ubuntu@ec2-18-223-151-11.us-east-2.compute.amazonaws...

```
banana:Desktop txing$ ssh -i "docker.pem" ubuntu@ec2-18-223-151-11.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-1047-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Nov  3 18:55:22 UTC 2019

  System load:  0.0               Processes:              92
  Usage of /:   57.4% of 7.69GB   Users logged in:        0
  Memory usage: 30%               IP address for eth0:    172.31.44.28
  Swap usage:   0%                IP address for docker0: 172.17.0.1

 * Congrats to the Kubernetes community on 1.16 beta 1! Now available
   in MicroK8s for evaluation and testing, with upgrades to RC and GA

     snap info microk8s

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

80 packages can be updated.
0 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


*** System restart required ***
Last login: Sun Nov  3 18:51:14 2019 from 98.109.123.76
ubuntu@ip-172-31-44-28:~$
```

# Download and Install Components

- On each VM (ARM and x86)
  - Install Docker

    ```
    $ sudo apt-get install
    $ sudo apt install docker.io
    ```

  - Download Popcorn CRIU

    ```
    $ git clone https://github.com/systems-nuts/criu.git
    $ git checkout heterogenous-simplified
    $ pip install ip_address
    $ pip install pyfastcopy
    ```

- Only on the x86 VM
  - Install Popcorn-compiler (not needed for this tutorial)

    ```
    $ git clone https://github.com/systems-nuts/popcorn-compiler.git
    ```

```
root@ip-172-31-16-7:~#  git clone https://github.com/systems-nuts/criu.git ; cd criu; git checkout heterogenous-simplified; pip install ip_address ; pip install pyfastcopy
Cloning into 'criu'...
Username for 'https://github.com': 123toorc@gmail.com
Password for 'https://123toorc@gmail.com@github.com':
remote: Enumerating objects: 354, done.
remote: Counting objects: 100% (354/354), done.
remote: Compressing objects: 100% (211/211), done.
remote: Total 56801 (delta 230), reused 233 (delta 132), pack-reused 56447
Receiving objects: 100% (56801/56801), 14.08 MiB | 13.55 MiB/s, done.
Resolving deltas: 100% (41426/41426), done.
error: pathspec 'heterogenous-simplified' did not match any file(s) known to git.
Collecting ip_address
  Downloading https://files.pythonhosted.org/packages/f3/8b/f6b7e28e8ebce256a385bc7790aa103617b56969cb70a08a76d5193ce832/ip_address-1.0.2.tar.gz
Collecting requests>=2.22.0 (from ip_address)
  Downloading https://files.pythonhosted.org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/requests-2.22.0-py2.py3-none-any.whl (57kB)
    100% |                                | 61kB 2.1MB/s
Requirement already satisfied: idna<2.9,>=2.5 in /usr/lib/python2.7/dist-packages (from requests>=2.22.0->ip_address)
Collecting chardet<3.1.0,>=3.0.2 (from requests>=2.22.0->ip_address)
  Downloading https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl (133kB)
    100% |                                | 143kB 3.4MB/s
Collecting certifi>=2017.4.17 (from requests>=2.22.0->ip_address)
  Downloading https://files.pythonhosted.org/packages/18/b0/8146a4f8dd402f60744fa380bc73ca47303cccf8b9190fd16a827281eac2/certifi-2019.9.11-py2.py3-none-any.whl (154kB)
    100% |                                | 163kB 4.2MB/s
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 (from requests>=2.22.0->ip_address)
  Downloading https://files.pythonhosted.org/packages/b4/40/a9837291310ee1ccc242ceb6ebfd9eb21539649f193a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl (125kB)
    100% |                                | 133kB 4.9MB/s
Building wheels for collected packages: ip-address
  Running setup.py bdist_wheel for ip-address ... done
  Stored in directory: /root/.cache/pip/wheels/72/ae/52/f52925ff32e5ef59392b620445198b5a6f9c633f1240c12f67
Successfully built ip-address
Installing collected packages: chardet, certifi, urllib3, requests, ip-address
Successfully installed certifi-2019.9.11 chardet-3.0.4 ip-address-1.0.2 requests-2.22.0 urllib3-1.25.7
Collecting pyfastcopy
  Downloading https://files.pythonhosted.org/packages/43/80/535d6b3de415e26d0a1cb774c6895dd07aa5986d2f8bde200393bd916790/pyfastcopy-1.0.3.tar.gz
Collecting mock (from pyfastcopy)
  Downloading https://files.pythonhosted.org/packages/05/d2/f94e68be6b17f46d2c353564da56e6fb89ef09faeeff3313a046cb810ca9/mock-3.0.5-py2.py3-none-any.whl
Collecting monotonic (from pyfastcopy)
  Downloading https://files.pythonhosted.org/packages/ac/aa/063eca6a416f397bd99552c534c6d11d57f58f2e94c14780f3bbf818c4cf/monotonic-1.5-py2.py3-none-any.whl
Collecting pysendfile (from pyfastcopy)
  Downloading https://files.pythonhosted.org/packages/cd/3f/4aa268afd0252f06b3b487c296a066a01ddd4222a46b7a3748599c8fc8c3/pysendfile-2.0.1.tar.gz
Collecting funcsigs>=1; python_version < "3.3" (from mock->pyfastcopy)
  Downloading https://files.pythonhosted.org/packages/69/cb/f5be453359271714c01b9bd06126eaf2e368f1fddfff30818754b5ac2328/funcsigs-1.0.2-py2.py3-none-any.whl
Requirement already satisfied: six in /usr/lib/python2.7/dist-packages (from mock->pyfastcopy)
Building wheels for collected packages: pyfastcopy, pysendfile
  Running setup.py bdist_wheel for pyfastcopy ... done
  Stored in directory: /root/.cache/pip/wheels/43/ef/5b/424484e3955a0ac0e364fbf2648aaaab4be135b69a8b2b216d
  Running setup.py bdist_wheel for pysendfile ... done
  Stored in directory: /root/.cache/pip/wheels/e5/2f/3f/d089a479eb292a361c50e998e7869d10e02b046664cfc67ecb
Successfully built pyfastcopy pysendfile
Installing collected packages: funcsigs, mock, monotonic, pysendfile, pyfastcopy
Successfully installed funcsigs-1.0.2 mock-3.0.5 monotonic-1.5 pyfastcopy-1.0.3 pysendfile-2.0.1
root@ip-172-31-16-7:~/criu#
```

# Docker Configuration #1

- Verify Docker is installed on each VM (official way from Docker)

  $ sudo docker run hello-world

```
Desktop — root@ip-172-31-46-231: ~ — ssh -i docker.pem ubuntu@ec2-18-188-53-61.us-east-2.compute.amazonaws.com — 96×29
root@ip-172-31-46-231:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

root@ip-172-31-46-231:~#
```

# Docker Configuration #2

- Turn on the experimental feature to enable dump and restore feature in Docker by editing */etc/docker/daemon.json*, if the file doesn't exist, create a new file (on each VM)

```
$ vim /etc/docker/daemon.json
```

  - Add the following lines

```
{
    "experimental": true
}
```

- Restart Docker (on each VM)

```
$ service docker restart
```

# CRIU Compilation

- CRIU dependencies (on each VM)

```
$ sudo apt-get update ; sudo apt-get install -y \
protobuf-c-compiler libprotobuf-c-dev gcc build-essential \
bsdmainutils python git-core asciidoc make htop \
curl supervisor cgroup-lite libapparmor-dev libseccomp-dev \
libprotobuf-dev libaio-dev apparmor libnet1-dev   \
protobuf-compiler python-protobuf libnl-3-dev libcap-dev
```

- CRIU compilation and install (on each VM)

```
$ cd criu
$ make ; sudo make install
```

```
root@ip-172-31-16-7:~# apt-get update ; apt-get install docker.io -y ; apt-get install protobuf-c-compiler libprotobuf-c-dev gcc build-essential bsdmainutils python git-core asciidoc make htop curl superv
isor cgroup-lite libapparmor-dev libseccomp-dev libprotobuf-dev libaio-dev apparmor libnet1-dev protobuf-compiler python-protobuf libnl-3-dev libcap-dev -y
Hit:1 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic InRelease
Get:2 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates InRelease [88.7 kB]
Get:3 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-backports InRelease [74.6 kB]
Get:4 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic/universe arm64 Packages [8,316 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports bionic-security InRelease [88.7 kB]
Get:6 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic/universe Translation-en [4,941 kB]
Get:7 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic/multiverse arm64 Packages [126 kB]
Get:8 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic/multiverse Translation-en [108 kB]
Get:9 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/main arm64 Packages [592 kB]
Get:10 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/main Translation-en [283 kB]
Get:11 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/restricted arm64 Packages [1,064 B]
Get:12 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/restricted Translation-en [6,128 B]
Get:13 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/universe arm64 Packages [917 kB]
Get:14 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/universe Translation-en [315 kB]
Get:15 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/multiverse arm64 Packages [2,752 B]
Get:16 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/multiverse Translation-en [4,044 B]
Get:17 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-backports/main arm64 Packages [2,512 B]
Get:18 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-backports/main Translation-en [1,644 B]
Get:19 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-backports/universe arm64 Packages [4,020 B]
Get:20 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-backports/universe Translation-en [1,856 B]
Get:21 http://ports.ubuntu.com/ubuntu-ports bionic-security/main arm64 Packages [385 kB]
Get:22 http://ports.ubuntu.com/ubuntu-ports bionic-security/main Translation-en [189 kB]
Get:23 http://ports.ubuntu.com/ubuntu-ports bionic-security/restricted arm64 Packages [668 B]
Get:24 http://ports.ubuntu.com/ubuntu-ports bionic-security/restricted Translation-en [4,292 B]
Get:25 http://ports.ubuntu.com/ubuntu-ports bionic-security/universe arm64 Packages [552 kB]
Get:26 http://ports.ubuntu.com/ubuntu-ports bionic-security/universe Translation-en [206 kB]
Get:27 http://ports.ubuntu.com/ubuntu-ports bionic-security/multiverse arm64 Packages [1,848 B]
Get:28 http://ports.ubuntu.com/ubuntu-ports bionic-security/multiverse Translation-en [2,568 B]
Fetched 17.2 MB in 5s (3,733 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils cgroupfs-mount containerd pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils cgroupfs-mount containerd docker.io pigz runc ubuntu-fan
0 upgraded, 7 newly installed, 0 to remove and 108 not upgraded.
Need to get 35.1 MB of archives.
After this operation, 202 MB of additional disk space will be used.
Get:1 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic/universe arm64 pigz arm64 2.4-1 [47.8 kB]
Get:2 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic/main arm64 bridge-utils arm64 1.5-15ubuntu1 [28.8 kB]
Get:3 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic/universe arm64 cgroupfs-mount all 1.4 [6,320 B]
Get:4 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/universe arm64 runc arm64 1.0.0~rc7+git20190403.029124da-0ubuntu1~18.04.2 [1,694 kB]
Get:5 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/universe arm64 containerd arm64 1.2.6-0ubuntu1~18.04.2 [13.6 MB]
Get:6 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic-updates/universe arm64 docker.io arm64 18.09.7-0ubuntu1~18.04.4 [19.7 MB]
Get:7 http://us-east-2.ec2.ports.ubuntu.com/ubuntu-ports bionic/main arm64 ubuntu-fan all 0.12.10 [34.7 kB]
Fetched 35.1 MB in 1s (49.3 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 59337 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.4-1_arm64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.5-15ubuntu1_arm64.deb ...
Unpacking bridge-utils (1.5-15ubuntu1) ...
Selecting previously unselected package cgroupfs-mount.
Preparing to unpack .../2-cgroupfs-mount_1.4_all.deb ...
Unpacking cgroupfs-mount (1.4) ...
```

```
root@ip-172-31-16-7:~/criu# vim /etc/docker/daemon.json
root@ip-172-31-16-7:~/criu# cat /etc/docker/daemon.json
{
        "experimental": true
}
root@ip-172-31-16-7:~/criu# service docker restart
root@ip-172-31-16-7:~/criu# make ; make install
sh: 1: pkg-config: not found
  GEN      .gitid
  GEN      criu/include/version.h
  GEN      include/common/asm
  PBCC     images/google/protobuf/descriptor.pb-c.c
  PBCC     images/opts.pb-c.c
  PBCC     images/sit.pb-c.c
  DEP      images/google/protobuf/descriptor.pb-c.d
  DEP      images/opts.pb-c.d
  DEP      images/sit.pb-c.d
  PBCC     images/macvlan.pb-c.c
  DEP      images/macvlan.pb-c.d
  PBCC     images/autofs.pb-c.c
  DEP      images/autofs.pb-c.d
  PBCC     images/sysctl.pb-c.c
  DEP      images/sysctl.pb-c.d
  PBCC     images/time.pb-c.c
  DEP      images/time.pb-c.d
  PBCC     images/binfmt-misc.pb-c.c
  DEP      images/binfmt-misc.pb-c.d
  PBCC     images/seccomp.pb-c.c
  DEP      images/seccomp.pb-c.d
  PBCC     images/userns.pb-c.c
  DEP      images/userns.pb-c.d
  PBCC     images/cgroup.pb-c.c
  DEP      images/cgroup.pb-c.d
  PBCC     images/fown.pb-c.c
  PBCC     images/ext-file.pb-c.c
  DEP      images/fown.pb-c.d
  DEP      images/ext-file.pb-c.d
  PBCC     images/rpc.pb-c.c
  DEP      images/rpc.pb-c.d
  PBCC     images/siginfo.pb-c.c
  DEP      images/siginfo.pb-c.d
  PBCC     images/pagemap.pb-c.c
  DEP      images/pagemap.pb-c.d
  PBCC     images/rlimit.pb-c.c
  DEP      images/rlimit.pb-c.d
  PBCC     images/file-lock.pb-c.c
  DEP      images/file-lock.pb-c.d
  PBCC     images/tty.pb-c.c
  DEP      images/tty.pb-c.d
  PBCC     images/tun.pb-c.c
```

# Check Installation is Working

- Make sure the installed CRIU is heterogenous version

```
root@ip-172-31-44-28:~# which criu-het
/usr/local/sbin/criu-het
root@ip-172-31-44-28:~# criu-het -V
Version: 3.11
GitID: 4f92d4ad
root@ip-172-31-44-28:~#
```

# Tutorial Material #1

- Download Popcorn H-container tutorial material from github

```
$ git clone https://github.com/systems-nuts/hcontainer-tutorial.git
```

# Tutorial Material #2

- **Contents**

```
root@ip-172-31-23-242:~# tree hcontainer-tutorial/
hcontainer-tutorial/
├── README.md
├── helloworld
│   ├── Dockerfile
│   ├── popcorn-hello
│   ├── popcorn-hello_aarch64
│   └── popcorn-hello_x86-64
├── popcorn-redis
│   ├── Dockerfile
│   ├── dump.rdb
│   ├── redis-server
│   ├── redis-server_aarch64
│   └── redis-server_x86-64
├── recode.sh
└── redis-benchmark
```

# Tutorial Material #3

- We provide examples step by step to explain different scenarios
  - From **ARM to x86**
    - Popcorn Hello-world migrates from ARM to x86 (Part 2)
  - From **x86 to ARM**
    - Popcorn Redis migrates from x86 to ARM (Part 3)
- Compilation process is exactly like Popcorn Linux
  - But you need to use the updated compiler

# Tutorial Part 2

## Hello World!

# Overview

1. About Docker
2. Create a Docker Image of Hello-world
3. Container Configuration of Hello-world
4. Het-Checkpoint of Hello-world
5. Make Sure Het-Checkpoint has been triggered for Hello-world
6. Convert dump files from aarch64 to x86_64 for Hello-world
7. Move Hello-world dump files to x86_64
8. Restore Hello-world on x86_64

# About Docker …

- Dockerfile
  - A text document that contains all the commands a user could call on the command line to assemble an image
- *docker build*
  - Users can create an automated build that executes several command-line instructions in succession
- *docker container*
  - Users can manage containers with this utility

# Create a Docker Image of Hello-world #1

- **On ARM only**
- Move to the application folder, and make a copy of the ARM binary removing the postfix specificating the architecture

```
$ cd /root/hcontainer-tutorial/helloworld
$ cp popcorn-hello_aarch64 popcorn-hello
```

- Copy the entire application directory in a directory called *app in / directory,* and create the docker image

```
$ cp -ar ../helloworld  /app
$ docker build –t helloworld .
$ docker container create helloworld
```

- It will return the id, for example

```
a40a7eb069172dc64dc771128cce91e942656f1cfe8b4d11ac97a99b08f64fd9
```

# Create a Docker Image of Hello-world #2

```
root@ip-172-31-23-242:~# cd hcontainer-tutorial/helloworld/
root@ip-172-31-23-242:~/hcontainer-tutorial/helloworld# cp popcorn-hello_aarch64  popcorn-hello
root@ip-172-31-23-242:~/hcontainer-tutorial/helloworld# uname -a
Linux ip-172-31-23-242 4.15.0-1047-aws #49-Ubuntu SMP Fri Aug 23 09:04:33 UTC 2019 aarch64 aarch64 aarch64 GNU/Linux
root@ip-172-31-23-242:~/hcontainer-tutorial/helloworld# cp -r ../helloworld/ /app
root@ip-172-31-23-242:~/hcontainer-tutorial/helloworld# docker build -t helloworld .
Sending build context to Docker daemon  27.69MB
Step 1/4 : FROM scratch
 --->
Step 2/4 : WORKDIR /app
 ---> Running in 3ea575a3b734
Removing intermediate container 3ea575a3b734
 ---> 25e43a5fafeb
Step 3/4 : COPY . /app
 ---> cb4e62e020ba
Step 4/4 : CMD ["./popcorn-hello"]
 ---> Running in ce556c8c1a3c
Removing intermediate container ce556c8c1a3c
 ---> 33ef5e3aaffa
Successfully built 33ef5e3aaffa
Successfully tagged helloworld:latest
root@ip-172-31-23-242:~/hcontainer-tutorial/helloworld# docker container create helloworld
bf8bc54e1debf283c9c5620bc3e966068dec4438a741cbb75a2149aaefd8fcaf
```

# Container Configuration of Hello-world #1

- **On ARM only**
- Configure the capabilities of the Container in order to allow all system calls, needed for Popcorn binary file

  ```
  $ sudo cd /var/lib/docker/containers/<$docker container ID>
  $ sudo vim hostconfig.json
  ```

  - Replace “CapAdd”: null with “CapAdd”: [“all”]
- Restart Docker service and start the Docker container

  ```
  $ service docker restart
  $ docker container start <$docker container ID>
  ```
- Make sure the Docker container is running with

  ```
  $ docker ps
  ```

# Container Configuration of Hello-world #2

txing — root@ip-172-31-23-242: /var/lib/docker/containers/bf8bc54e1debf283c9c5620bc3e966068dec4438a741cbb75a2149aaefd8fcaf — ssh ◂ -bash — 110×41

```
root@ip-172-31-23-242:/var/lib/docker/containers/bf8bc54e1debf283c9c5620bc3e966068dec4438a741cbb75a2149aaefd8f
caf# cat hostconfig.json  | grep "CapAdd"
{"Binds":null,"ContainerIDFile":"","LogConfig":{"Type":"json-file","Config":{}},"NetworkMode":"default","PortB
indings":{},"RestartPolicy":{"Name":"no","MaximumRetryCount":0},"AutoRemove":false,"VolumeDriver":"","VolumesF
rom":null,"CapAdd":["all"],"CapDrop":null,"Dns":[],"DnsOptions":[],"DnsSearch":[],"ExtraHosts":null,"GroupAdd"
:null,"IpcMode":"shareable","Cgroup":"","Links":null,"OomScoreAdj":0,"PidMode":"","Privileged":false,"PublishA
llPorts":false,"ReadonlyRootfs":false,"SecurityOpt":null,"UTSMode":"","UsernsMode":"","ShmSize":67108864,"Runt
ime":"runc","ConsoleSize":[0,0],"Isolation":"","CpuShares":0,"Memory":0,"NanoCpus":0,"CgroupParent":"","BlkioW
eight":0,"BlkioWeightDevice":[],"BlkioDeviceReadBps":null,"BlkioDeviceWriteBps":null,"BlkioDeviceReadIOps":nul
l,"BlkioDeviceWriteIOps":null,"CpuPeriod":0,"CpuQuota":0,"CpuRealtimePeriod":0,"CpuRealtimeRuntime":0,"CpusetC
pus":"","CpusetMems":"","Devices":[],"DeviceCgroupRules":null,"DiskQuota":0,"KernelMemory":0,"MemoryReservatio
n":0,"MemorySwap":0,"MemorySwappiness":null,"OomKillDisable":false,"PidsLimit":0,"Ulimits":null,"CpuCount":0,"
CpuPercent":0,"IOMaximumIOps":0,"IOMaximumBandwidth":0,"MaskedPaths":["/proc/asound","/proc/acpi","/proc/kcore
","/proc/keys","/proc/latency_stats","/proc/timer_list","/proc/timer_stats","/proc/sched_debug","/proc/scsi","
/sys/firmware"],"ReadonlyPaths":["/proc/bus","/proc/fs","/proc/irq","/proc/sys","/proc/sysrq-trigger"]}
root@ip-172-31-23-242:/var/lib/docker/containers/bf8bc54e1debf283c9c5620bc3e966068dec4438a741cbb75a2149aaefd8f
caf#
```

# Het-Checkpoint of Hello-world #1

- **On ARM only**
- Prepare for heterogeneous checkpoint, which require to notify the heterogeneous-ready binary that a migration will happen
  - 2 arguments: first process ID, second target ISA

```
$ PID=`ps –A | grep popcorn-hello`
$ popcorn-notify $PID x86-64
```

- (Normal) Docker checkpoint
  - The latest version od Docker uses pipes to integrate with CRIU, this instruction will create dump images of target container

```
$ docker checkpoint create <$docker container ID> mycheckpoint
```

  - *mycheckpoint* can be any name

# Het-Checkpoint of Hello-world #2

txing — root@ip-172-31-23-242: ~ — ssh ◂ -bash — 110×41

```
root@ip-172-31-23-242:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
          NAMES
bf8bc54e1deb        helloworld          "./popcorn-hello"   3 days ago          Up 15 seconds
          keen_perlman
root@ip-172-31-23-242:~# ps -A | grep popcorn-hello
 7082 ?        00:00:00 popcorn-hello
root@ip-172-31-23-242:~# popcorn-notify  7082 x86-64
argv2: x86-64
argv3: (null)
get_binary_path: proc exec path is /proc/7082/exe
The process stopped a first time 7082, 8010f0
putdata addr pid 7082 8010f0 data 1
stopped by signal 5
cont with sig 0
cont sent with sig 0
going to wait 0
ret data -1
root@ip-172-31-23-242:~# docker checkpoint create bf8bc54e1deb mycheckpoint
mycheckpoint
root@ip-172-31-23-242:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
          NAMES
root@ip-172-31-23-242:~#
```

# Make Sure Het-Checkpoint Has Been Triggered for Hello-world

- **On ARM only**
- Check the output log of the container

  $ cat /var/lib/docker/containers/<$docker container ID>-json.log

```
root@ip-172-31-23-242:~/hcontainer-tutorial/helloworld# cat /var/lib/docker/containers/bf8bc54e1debf283c9c5620bc3e966068dec4438a741cbb75a2149aaefd8fcaf/bf8bc54e1debf283c9c5620bc3e966068dec4438a741cbb75a21
49aaefd8fcaf-json.log
{"log":"[1] iteration 0\n","stream":"stdout","time":"2019-10-30T21:03:08.297166036Z"}
{"log":"[1] iteration 1\n","stream":"stdout","time":"2019-10-30T21:04:03.302919422Z"}
{"log":"[1] iteration 2\n","stream":"stdout","time":"2019-10-30T21:04:03.302979314Z"}
{"log":"[1] iteration 3\n","stream":"stdout","time":"2019-10-30T21:04:03.30299471Z"}
{"log":"[1] iteration 4\n","stream":"stdout","time":"2019-10-30T21:04:03.303007718Z"}
{"log":"[1] iteration 5\n","stream":"stdout","time":"2019-10-30T21:04:03.303020102Z"}
{"log":"[1] iteration 6\n","stream":"stdout","time":"2019-10-30T21:04:03.303051458Z"}
{"log":"[1] iteration 7\n","stream":"stdout","time":"2019-10-30T21:04:03.30306641Z"}
{"log":"[1] iteration 42\n","stream":"stdout","time":"2019-10-30T21:04:03.303610529Z"}
{"log":"[1] iteration 43\n","stream":"stdout","time":"2019-10-30T21:04:03.303641453Z"}
{"log":"[1] iteration 44\n","stream":"stdout","time":"2019-10-30T21:04:03.303655145Z"}
{"log":"[1] iteration 45\n","stream":"stdout","time":"2019-10-30T21:04:03.303666965Z"}
{"log":"dest arch is 1\n","stream":"stdout","time":"2019-10-30T21:04:03.303678653Z"}
{"log":"__migrate_shim_internal 121\n","stream":"stdout","time":"2019-10-30T21:04:03.303690281Z"}
{"log":"__migrate_shim_internal 123\n","stream":"stdout","time":"2019-10-30T21:04:03.305172132Z"}
{"log":"__migrate_shim_internal raising done 134\n","stream":"stdout","time":"2019-10-30T21:04:03.305195148Z"}
{"log":"[1] iteration 46\n","stream":"stdout","time":"2019-10-30T21:04:03.305224848Z"}
{"log":"[1] iteration 47\n","stream":"stdout","time":"2019-10-30T21:04:03.305239992Z"}
{"log":"[1] iteration 48\n","stream":"stdout","time":"2019-10-30T21:04:03.3052512Z"}
{"log":"[1] iteration 49\n","stream":"stdout","time":"2019-10-30T21:04:03.305262216Z"}
```

# Convert dump files from aarch64 to x86_64 for Hello-world #1

- **On ARM only**
- Recode the aarch64 dump files to x86_64 dump files
- Use the *recode.sh* script
  - 3 arguments
    - $<docker container ID>
    - $checkpoint name
    - $target ISA
  - Example

```
./recode.sh a40a7eb069172dc64dc771128cce91e942656f1cfe8b4d11ac97a99b08f64fd9 mycheckpoint x86-64
```

# Convert dump files from aarch64 to x86_64 for Hello-world #2

```
root@ip-172-31-23-242:~/hcontainer-tutorial# ./recode.sh bf8bc54e1debf283c9c5620bc3e966068dec4438a741cbb75a2149aaefd8fcaf mycheckpoint x86-64
(1, 0.0144231319427490 23, 0.028733015060424805, 7.796287536621094e-05, 0.001973867416381836, 0.028670072555541992)
/root/hcontainer-tutorial
root@ip-172-31-23-242:~/hcontainer-tutorial# ls
README.md  helloworld  mycheckpoint  popcorn-redis  recode.sh  redis-benchmark
```

# Move Hello-world dump files to x86_64 #1

- **On ARM only**
- Copy the recoded checkpoint directory to the target machine

  ```
  $ scp –r mycheckpoint root@target_ip:/root
  ```

- Make sure the copy was successful

# Move Hello-world dump files to x86_64 #2

```
root@ip-172-31-23-242:~/hcontainer-tutorial# scp -r mycheckpoint/ root@18.223.151.11:~
tmpfs-dev-60.tar.gz.img                                     100%  363   455.3KB/s   00:00
seccomp.img                                                 100% 5788     5.9MB/s   00:00
fdinfo-2.img                                                100%   68   103.1KB/s   00:00
utsns-11.img                                                100%   34    52.1KB/s   00:00
pages-1.img                                                 100%  232KB  28.5MB/s   00:00
pagemap-1.img                                               100%  216   353.8KB/s   00:00
fs-1.img                                                    100%   18    29.9KB/s   00:00
descriptors.json                                            100%   45    77.5KB/s   00:00
pstree.img                                                  100%   22    36.4KB/s   00:00
files.img                                                   100%  536   799.2KB/s   00:00
core-1.img                                                  100% 1776     2.5MB/s   00:00
tmpfs-dev-63.tar.gz.img                                     100%   98   160.2KB/s   00:00
mm-1.img                                                    100%  572   917.5KB/s   00:00
tmpfs-dev-61.tar.gz.img                                     100%   98   156.5KB/s   00:00
inventory.img                                               100%   42    72.4KB/s   00:00
tmpfs-dev-57.tar.gz.img                                     100%  385   634.3KB/s   00:00
ipcns-var-10.img                                            100%   82   109.5KB/s   00:00
ids-1.img                                                   100%   34    51.6KB/s   00:00
cgroup.img                                                  100% 5010     6.3MB/s   00:00
tmpfs-dev-62.tar.gz.img                                     100%   98   169.6KB/s   00:00
mountpoints-12.img                                          100% 4192     5.4MB/s   00:00
root@ip-172-31-23-242:~/hcontainer-tutorial# ssh root@18.223.151.11
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-1047-aws x86_64)
```

```
Last login: Wed Oct 30 20:42:09 2019 from 18.217.14.164
root@ip-172-31-44-28:~# uname -a
Linux ip-172-31-44-28 4.15.0-1047-aws #49-Ubuntu SMP Fri Aug 23 09:02:45 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
root@ip-172-31-44-28:~# ls
criu  hcontainer-tutorial  mycheckpoint
root@ip-172-31-44-28:~#
```

# Restore Hello-world on x86_64 #1

- Create a Docker image on x86_64 (could have been done before)
  - Same instruction as in aarch64, but

```
$ cp popcorn-hello_x86-64 popcorn-hello
```

- Restore the checkpoint

```
$ sudo cp –r /root/mycheckpoint /var/lib/docker/containers/$<docker container id>/checkpoints/
$ docker container start --checkpoint mycheckpoint <$docker container id>
$ docker ps
```

# Restore Hello-world on x86_64 #2

```
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# ls
Dockerfile  popcorn-hello  popcorn-hello_aarch64  popcorn-hello_x86-64
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# cp popcorn-hello_x86-64  popcorn-hello
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# cp -r ../helloworld/ /app
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# uname -a
Linux ip-172-31-44-28 4.15.0-1047-aws #49-Ubuntu SMP Fri Aug 23 09:02:45 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# docker build -t x86hello .
Sending build context to Docker daemon  28.65MB
Step 1/4 : FROM scratch
 --->
Step 2/4 : WORKDIR /app
 ---> Running in 3a6310debd26
Removing intermediate container 3a6310debd26
 ---> 9690db2b5c5a
Step 3/4 : COPY . /app
 ---> cac8c2a6d159
Step 4/4 : CMD ["./popcorn-hello"]
 ---> Running in 89df85c72c2c
Removing intermediate container 89df85c72c2c
 ---> a46d491ac8ec
Successfully built a46d491ac8ec
Successfully tagged x86hello:latest
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# docker container create x86hello
995cfcebda6905b919b6642e20b850a60018b9ee0aeed6afe153159d35281742
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# vim /var/lib/docker/containers/995cfcebda6905b919b6642e20b850a60018b9ee0aeed6afe153159d35281742/hostconfig.json
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# cat /var/lib/docker/containers/995cfcebda6905b919b6642e20b850a60018b9ee0aeed6afe153159d35281742/hostconfig.json |grep CapAdd
{"Binds":null,"ContainerIDFile":"","LogConfig":{"Type":"json-file","Config":{}},"NetworkMode":"default","PortBindings":{},"RestartPolicy":{"Name":"no","MaximumRetryCount":0},"AutoRemove":false,"VolumeDriv
er":"","VolumesFrom":null,"CapAdd":["all"],"CapDrop":null,"Dns":[],"DnsOptions":[],"DnsSearch":[],"ExtraHosts":null,"GroupAdd":null,"IpcMode":"shareable","Cgroup":"","Links":null,"OomScoreAdj":0,"PidMode"
:"","Privileged":false,"PublishAllPorts":false,"ReadonlyRootfs":false,"SecurityOpt":null,"UTSMode":"","UsernsMode":"","ShmSize":67108864,"Runtime":"runc","ConsoleSize":[0,0],"Isolation":"","CpuShares":0,"
Memory":0,"NanoCpus":0,"CgroupParent":"","BlkioWeight":0,"BlkioWeightDevice":[],"BlkioDeviceReadBps":null,"BlkioDeviceWriteBps":null,"BlkioDeviceReadIOps":null,"BlkioDeviceWriteIOps":null,"CpuPeriod":0,"C
puQuota":0,"CpuRealtimePeriod":0,"CpuRealtimeRuntime":0,"CpusetCpus":"","CpusetMems":"","Devices":[],"DeviceCgroupRules":null,"DiskQuota":0,"KernelMemory":0,"MemoryReservation":0,"MemorySwap":0,"MemorySwa
ppiness":null,"OomKillDisable":false,"PidsLimit":0,"Ulimits":null,"CpuCount":0,"CpuPercent":0,"IOMaximumIOps":0,"IOMaximumBandwidth":0,"MaskedPaths":["/proc/asound","/proc/acpi","/proc/kcore","/proc/keys"
,"/proc/latency_stats","/proc/timer_list","/proc/timer_stats","/proc/sched_debug","/proc/scsi","/sys/firmware"],"ReadonlyPaths":["/proc/bus","/proc/fs","/proc/irq","/proc/sys","/proc/sysrq-trigger"]}
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# service docker restart
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# cp -r ~/mycheckpoint/ /var/lib/docker/containers/995cfcebda6905b919b6642e20b850a60018b9ee0aeed6afe153159d35281742/checkpoints/
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# docker container start --checkpoint mycheckpoint 995cfcebda6905b919b6642e20b850a60018b9ee0aeed6afe153159d35281742
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
995cfcebda69        x86hello            "./popcorn-hello"   About a minute ago   Up 3 seconds                            distracted_raman
root@ip-172-31-44-28:~/hcontainer-tutorial/helloworld#
```

# Tutorial Part 3

## Redis, a Real-world Application

# Overview

1. Start Redis on x86_64
2. Test Redis functionality on x86_64
3. Het-Checkpoint Redis on x86_64
4. Restore Redis on aarch64
5. Test Redis Functionality on aarch64

# Start Redis on x86_64 #1

- Prepare Dockerfile
  - In dockerfile, make sure to have the argument “--protected-mode”, “no”
  - Otherwise Redis server will reject all requests except localhost
- Create Docker image

```
$ cd /root/hcontainer-tutorial/popcorn-redis
$ cp popcorn-redis_x86-64 popcorn-redis
$ cp –r ../popcorn-redis    /app
$ docker build –t popcorn-redis .
```

- Run Docker image
  - (Differently from before we will use docker run with some argument, additionally try to modify hostconfig.json file as we did in helloworld)

```
$ docker run --cap-add all -d-p 6379:6379 popcorn-redis
```

    - -d (detach which will create a container)
    - -p (map host port to container port)
    - -cap-add all (add all capabilities)
    - It will return a container id and it was already running

# Start Redis on x86_64 #2

Desktop — root@ip-172-31-44-28: ~/hcontainer-tutorial/popcorn-redis — ssh ◂ -bash — 130×48

```
root@ip-172-31-44-28:~# uname -a
Linux ip-172-31-44-28 4.15.0-1047-aws #49-Ubuntu SMP Fri Aug 23 09:02:45 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
root@ip-172-31-44-28:~# cd hcontainer-tutorial/popcorn-redis/
root@ip-172-31-44-28:~/hcontainer-tutorial/popcorn-redis# cp redis-server_x86-64  redis-server
root@ip-172-31-44-28:~/hcontainer-tutorial/popcorn-redis# rm -r /app ; cp -r ../popcorn-redis/ /app
root@ip-172-31-44-28:~/hcontainer-tutorial/popcorn-redis# docker build -t popcorn-redis .
Sending build context to Docker daemon  37.72MB
Step 1/4 : FROM ubuntu:18.04
 ---> cf0f3ca922e0
Step 2/4 : WORKDIR /app
 ---> Using cache
 ---> 7f73af15d992
Step 3/4 : COPY . /app
 ---> Using cache
 ---> 7378388e3f4e
Step 4/4 : CMD ["./redis-server","--protected-mode","no"]
 ---> Using cache
 ---> 710f1200118c
Successfully built 710f1200118c
Successfully tagged popcorn-redis:latest
root@ip-172-31-44-28:~/hcontainer-tutorial/popcorn-redis# docker run --cap-add all -d -p 6379:6379 popcorn-redis
7c18d0e62db14d3ab567ee27d33b514d3e373eb7ce91e4e6bf24ee056da7eb94
root@ip-172-31-44-28:~/hcontainer-tutorial/popcorn-redis# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
7c18d0e62db1        popcorn-redis       "./redis-server --pr…"   17 seconds ago      Up 16 seconds       0.0.0.0:6379->6379/tcp
stoic_fermat
root@ip-172-31-44-28:~/hcontainer-tutorial/popcorn-redis#
```

# Test Redis Functionality on x86_64 #1

- In order to later check that migration is correct
- Use *redis-cli* to add a special key-value to Redis that we will test after migration
  - Command to be used

```
$ redis-cli  -h $ip -p $port_num
```

  - Example

```
$ redis-cli -h 127.0.0.1 -p 6379
127.0.0.1 >  get popcorn
(nil)
127.0.0.1 >  add popcorn go
OK
127.0.0.1 >  get popcorn
“go”
```

# Test Redis Functionality on x86_46 #2

```
root@ip-172-31-44-28:~# redis-cli
127.0.0.1:6379> get popcorn
(nil)
127.0.0.1:6379> set popcorn go
OK
127.0.0.1:6379> get popcorn
"go"
127.0.0.1:6379> exit
root@ip-172-31-44-28:~#
```

# Het-Checkpoint Redis on x86_64 #1

- Create a Heterogeneous Docker checkpoint

```
$ PID=`ps –A | awk ‘/redis/{print $1}’`
$ popcorn-notify $PID aarch64
$ docker checkpoint create <$docker container id>  redis-check
```

  - Where *redis-check* is *<$checkpoint name>*
- Recode the checkpoint dump files with
  - *./recode <$docker container ID> <$checkpoint name> <$target ISA>*
  - Example

```
$ ./recode <$docker container ID> redis-check aarch64
```

- Copy the dump files to the target ISA by using *scp*

# Het-Checkpoint Redis on x86_64 #2

Desktop — root@ip-172-31-44-28: ~/hcontainer-tutorial — ssh ◂ -bash — 130×48

```
root@ip-172-31-44-28:~# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
7c18d0e62db1        popcorn-redis       "./redis-server --pr…"   2 minutes ago       Up 2 minutes        0.0.0.0:6379->6379/tcp
stoic_fermat
root@ip-172-31-44-28:~# ps -A | grep redis
16070 ?        00:00:00 redis-server
root@ip-172-31-44-28:~# popcorn-notify 16070 aarch64
argv2: aarch64
argv3: (null)
get_binary_path: proc exec path is /proc/16070/exe
The process stopped a first time 16070, 805da0
putdata addr pid 16070 805da0 data 0
stopped by signal 5
cont with sig 0
cont sent with sig 0
going to wait 0
ret data -1
root@ip-172-31-44-28:~# docker checkpoint create 7c18d0e62db1 redis-check
redis-check
root@ip-172-31-44-28:~# ls
criu  hcontainer-tutorial  redis
```

# Het-Checkpoint Redis on x86_64 #2

```
Desktop — root@ip-172-31-44-28: ~/hcontainer-tutorial — ssh ◂ -bash — 135×51
root@ip-172-31-44-28:~/hcontainer-tutorial# ls
README.md  helloworld  popcorn-redis  recode.sh  redis-benchmark  redis-cli
root@ip-172-31-44-28:~/hcontainer-tutorial# ./recode.sh 7c18d0e62db14d3ab567ee27d33b514d3e373eb7ce91e4e6bf24ee056da7eb94 redis-check aa
rch64
+ set -e
+ PID=7c18d0e62db14d3ab567ee27d33b514d3e373eb7ce91e4e6bf24ee056da7eb94
+ CHECKPOINT_NAME=redis-check
+ TARGET=aarch64
+ mkdir /var/lib/docker/containers/7c18d0e62db14d3ab567ee27d33b514d3e373eb7ce91e4e6bf24ee056da7eb94/checkpoints/redis-check/simple
+ cp /var/lib/docker/containers/7c18d0e62db14d3ab567ee27d33b514d3e373eb7ce91e4e6bf24ee056da7eb94/checkpoints/redis-check/descriptors.js
on /var/lib/docker/containers/7c18d0e62db14d3ab567ee27d33b514d3e373eb7ce91e4e6bf24ee056da7eb94/checkpoints/redis-check/simple
+ cd /var/lib/docker/containers/7c18d0e62db14d3ab567ee27d33b514d3e373eb7ce91e4e6bf24ee056da7eb94/checkpoints/redis-check/
+ crit recode -t aarch64 -o simple
(1, 0.007384061813354492, 0.04302692413330078, 5.817413330078125e-05, 0.0057849884033203125, 0.006586790084838867)
+ cd -
/root/hcontainer-tutorial
+ cp -r /var/lib/docker/containers/7c18d0e62db14d3ab567ee27d33b514d3e373eb7ce91e4e6bf24ee056da7eb94/checkpoints/redis-check/simple ./re
dis-check
+ rm -r /var/lib/docker/containers/7c18d0e62db14d3ab567ee27d33b514d3e373eb7ce91e4e6bf24ee056da7eb94/checkpoints/redis-check/simple
+ for i in ./$CHECKPOINT_NAME/core-*
+ crit decode -i ./redis-check/core-1.img -o ./redis-check/core-1.img.dec
+ sed -i 's#"seccomp_mode": "filter",# #' ./redis-check/core-1.img.dec
+ crit encode -i ./redis-check/core-1.img.dec -o ./redis-check/core-1.img
+ rm ./redis-check/core-1.img.dec
root@ip-172-31-44-28:~/hcontainer-tutorial# ls
README.md  helloworld  popcorn-redis  recode.sh  redis-benchmark  redis-check  redis-cli
root@ip-172-31-44-28:~/hcontainer-tutorial# scp -r redis-check/ arm
root@ip-172-31-44-28:~/hcontainer-tutorial# scp -r redis-check/ arm:~
fdinfo-2.img                                  100%  128   174.0KB/s   00:00
core-1.img                                    100% 1520     2.1MB/s   00:00
tmpfs-dev-65.tar.gz.img                       100%   98   154.0KB/s   00:00
cgroup.img                                    100% 5010     6.3MB/s   00:00
pagemap-1.img                                 100% 1427     2.1MB/s   00:00
ids-1.img                                     100%   34    55.6KB/s   00:00
fs-1.img                                      100%   18    30.6KB/s   00:00
pages-1.img                                   100% 1960KB  66.2MB/s   00:00
inventory.img                                 100%   42    68.7KB/s   00:00
ipcns-var-10.img                              100%   82   144.7KB/s   00:00
tmpfs-dev-63.tar.gz.img                       100%  367   596.9KB/s   00:00
```

# Restore Redis on aarch64 #1

- Create Docker Redis image as we did before (but for aarch64)
- Run Docker Redis image as we did before (but for aarch64)
- Stop the container, because docker didn't offer port mapping flag when create a container

  ```
  $ docker container stop <$container ID>
  ```

- Copy the checkpoint directory we received from x86

  ```
  $ cp -r <$checkpoint name> /var/lib/docker/containers/<$container id>/checkpoints/
  ```

- Restore the container

  ```
  $ docker container start --checkpoint <$checkpoint name>  <$container id>
  ```

# Restore Redis on aarch64 #2

```
root@ip-172-31-23-242:~# uname -a
Linux ip-172-31-23-242 4.15.0-1047-aws #49-Ubuntu SMP Fri Aug 23 09:04:33 UTC 2019 aarch64 aarch64 aarch64 GNU/Linux
root@ip-172-31-23-242:~# ls
criu  hcontainer-tutorial  redis-check
root@ip-172-31-23-242:~# cd hcontainer-tutorial/popcorn-redis/
root@ip-172-31-23-242:~/hcontainer-tutorial/popcorn-redis# cp redis-server_aarch64 redis-server
root@ip-172-31-23-242:~/hcontainer-tutorial/popcorn-redis# ls
Dockerfile  redis-server  redis-server_aarch64  redis-server_x86-64
root@ip-172-31-23-242:~/hcontainer-tutorial/popcorn-redis# docker build -t myredis
"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
root@ip-172-31-23-242:~/hcontainer-tutorial/popcorn-redis# docker build -t myredis .
Sending build context to Docker daemon  37.08MB
Step 1/4 : FROM ubuntu:18.04
18.04: Pulling from library/ubuntu
817f52ea6299: Pull complete
115b95df9208: Pull complete
9b2d56ba49a4: Pull complete
b92cf01ea129: Pull complete
Digest: sha256:a7b8b7b33e44b123d7f997bd4d3d0a59fafc63e203d17efedf09ff3f6f516152
Status: Downloaded newer image for ubuntu:18.04
 ---> 059b1dd9b693
Removing intermediate container 1a396fb66f56
 ---> 85678c3ea821
Successfully built 85678c3ea821
Successfully tagged myredis:latest
root@ip-172-31-23-242:~/hcontainer-tutorial/popcorn-redis# cd ..
root@ip-172-31-23-242:~/hcontainer-tutorial# cp -r popcorn-redis/ /app
root@ip-172-31-23-242:~/hcontainer-tutorial# docker run --cap-add all -d -p 6379:6379 myredis
9c02d0b1caf6c6cf66a13ced958a617dc9c5d4f36f6b3dd55168969fb543b96f
root@ip-172-31-23-242:~/hcontainer-tutorial# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
9c02d0b1caf6        myredis             "./redis-server --pr…"   3 seconds ago       Up 1 second         0.0.0.0:6379->6379/tcp   kind_tu
root@ip-172-31-23-242:~/hcontainer-tutorial# docker container stop  9c02d0b1caf6c6cf66a13ced958a617dc9c5d4f36f6b3dd55168969fb543b96f
9c02d0b1caf6c6cf66a13ced958a617dc9c5d4f36f6b3dd55168969fb543b96f
```

# Restore Redis on aarch64 #3

```
root@ip-172-31-23-242:~/hcontainer-tutorial# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
root@ip-172-31-23-242:~/hcontainer-tutorial# ls
README.md  helloworld  mycheckpoint  popcorn-redis  recode.sh  redis-benchmark
root@ip-172-31-23-242:~/hcontainer-tutorial# cd ..
root@ip-172-31-23-242:~# ls
criu  hcontainer-tutorial  redis-check
root@ip-172-31-23-242:~# cp -r redis-check/ /var/lib/docker/containers/9c02d0b1caf6c6cf66a13ced958a617dc9c5d4f36f6b3dd55168969fb543b96f/checkpoints/
root@ip-172-31-23-242:~# docker container start --checkpoint redis-check 9c02d0b1caf6c6cf66a13ced958a617dc9c5d4f36f6b3dd55168969fb543b96f
root@ip-172-31-23-242:~# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
9c02d0b1caf6        myredis             "./redis-server --pr…"   About a minute ago   Up 3 seconds        0.0.0.0:6379->6379/tcp   kind_tu
root@ip-172-31-23-242:~# exit
logout
Connection to 18.217.14.164 closed.
```

# Test Redis Functionality on aarch64 #1

- Check that Redis has been restored correctly
  - Use *redis-cli*

```
$ redis-cli -h 127.0.0.1
127.0.0.1 >  get popcorn
“go”
```

# Test Redis Functionality on aarch64 #2

```
root@ip-172-31-44-28:~/hcontainer-tutorial# uname -a
Linux ip-172-31-44-28 4.15.0-1047-aws #49-Ubuntu SMP Fri Aug 23 09:02:45 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
root@ip-172-31-44-28:~/hcontainer-tutorial# redis-cli -h 18.217.14.164
18.217.14.164:6379> get popcorn
"go"
18.217.14.164:6379>
```

Running redis-cli on X86 to connect to ARM machine check the key-value we set before was success migrated.

abarbala@[stevens.edu|ed.ac.uk]
txing1@stevens.edu
binoy@vt.edu

**www.popcornlinux.org**

# Thanks!

## Q&A