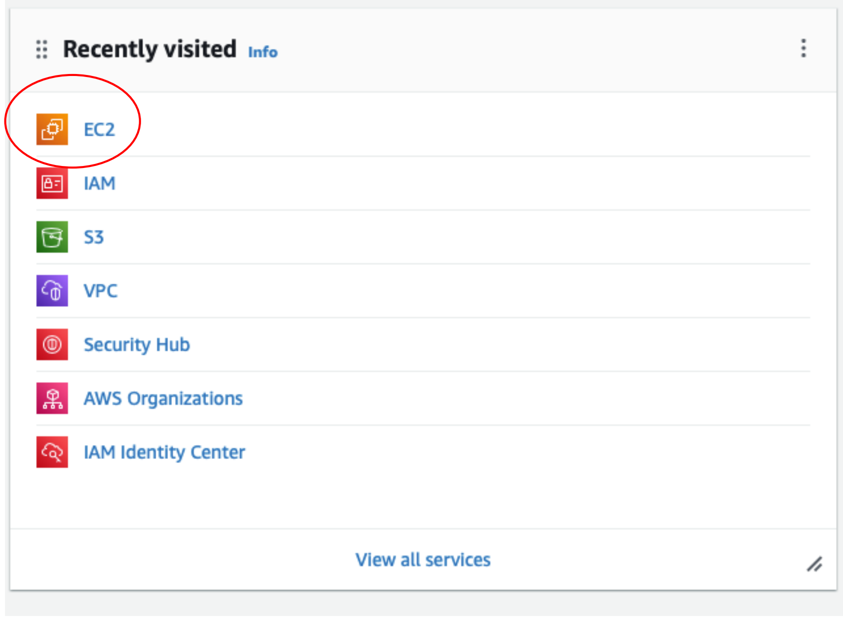
FPGA Designs on AWS

Programming FPGAs for Economics:
An Introduction to Electrical Engineering Economics

Bhagath Cheela, Alessandro Peri, André DeHon, Jesús Fernández-Villaverde

Steps

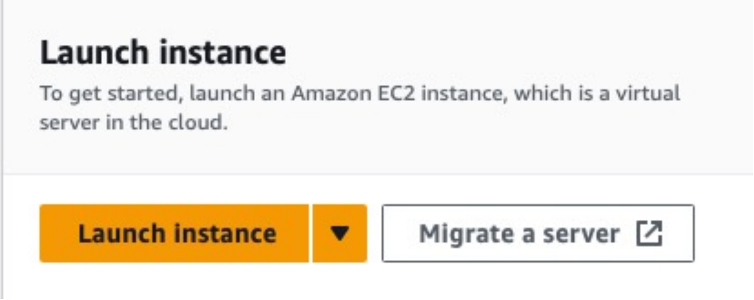


Recently visited Info

- EC2
- IAM
- S3
- VPC
- Security Hub
- AWS Organizations
- IAM Identity Center

View all services

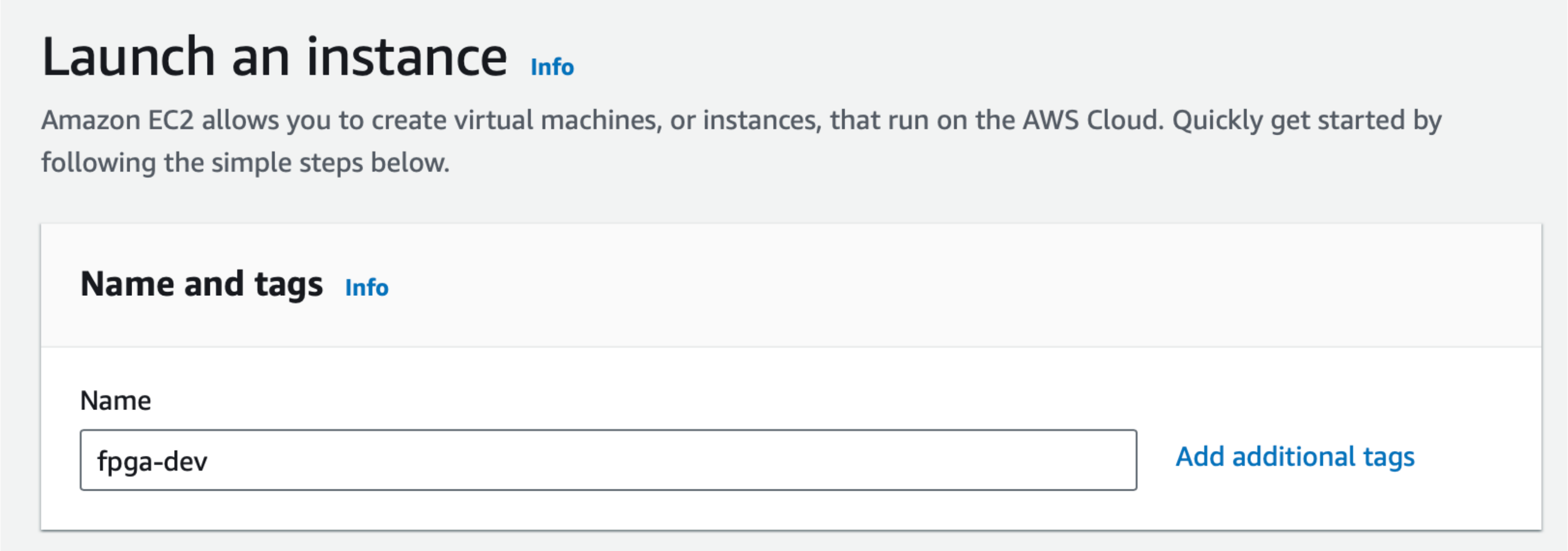
1. **Log into your AWS account:**
2. Navigate to the Home Console
3. Select **EC2**
4. Launch Instance



Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Steps: Name and tags



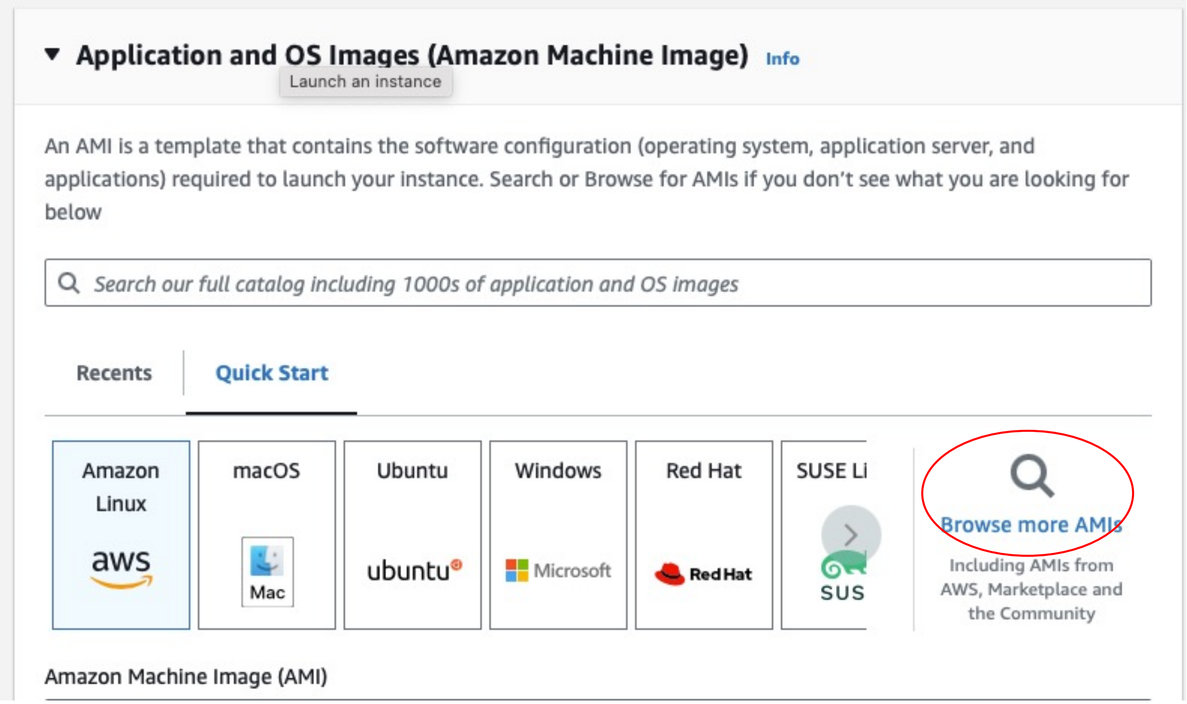
Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: fpga-dev

Select FPGA Developer AMI: Browse more AMI



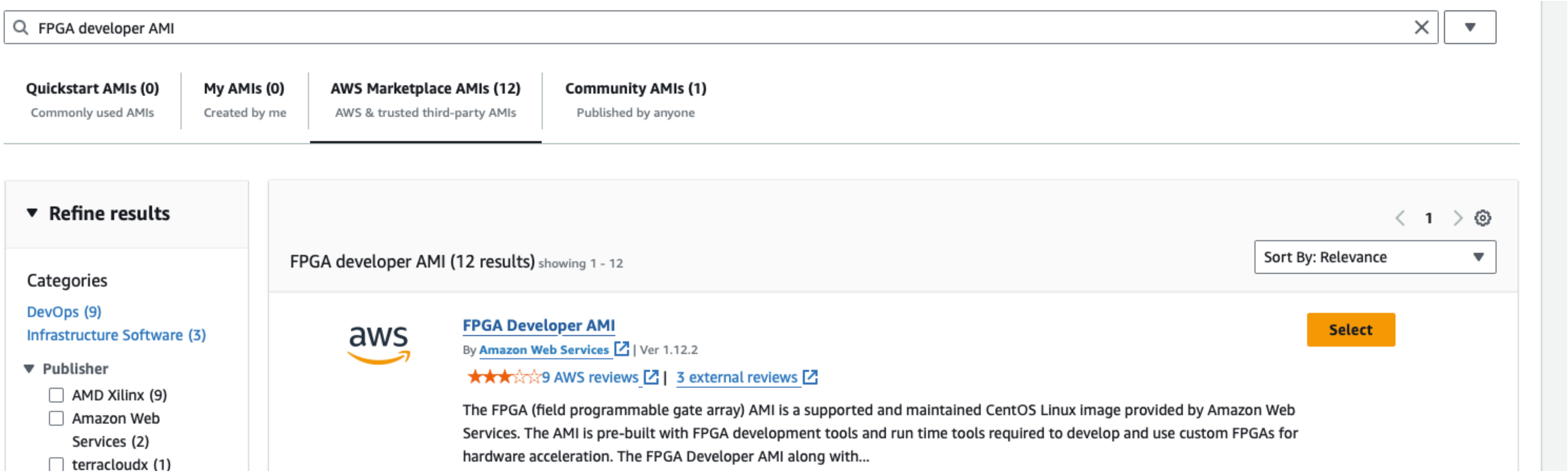
▼ **Application and OS Images (Amazon Machine Image)** Info

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

Search our full catalog including 1000s of application and OS images

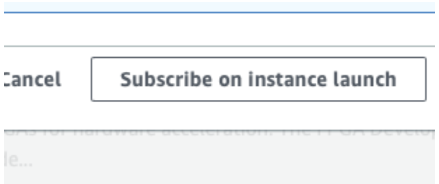
Amazon Machine Image (AMI)

Select FPGA Developer AMI

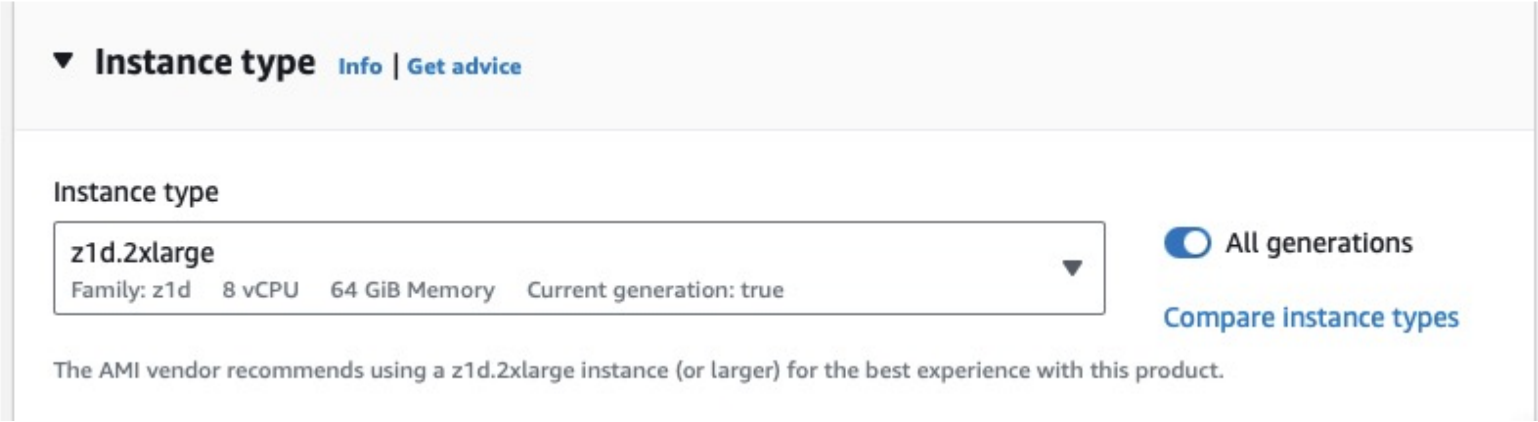


The FPGA (field programmable gate array) AMI is a supported and maintained CentOS Linux image provided by Amazon Web Services. The AMI is pre-built with FPGA development tools and run time tools required to develop and use custom FPGAs for hardware acceleration. The FPGA Developer AMI along with...

Select: Subscribe on instance Launch.



Select Build Instance

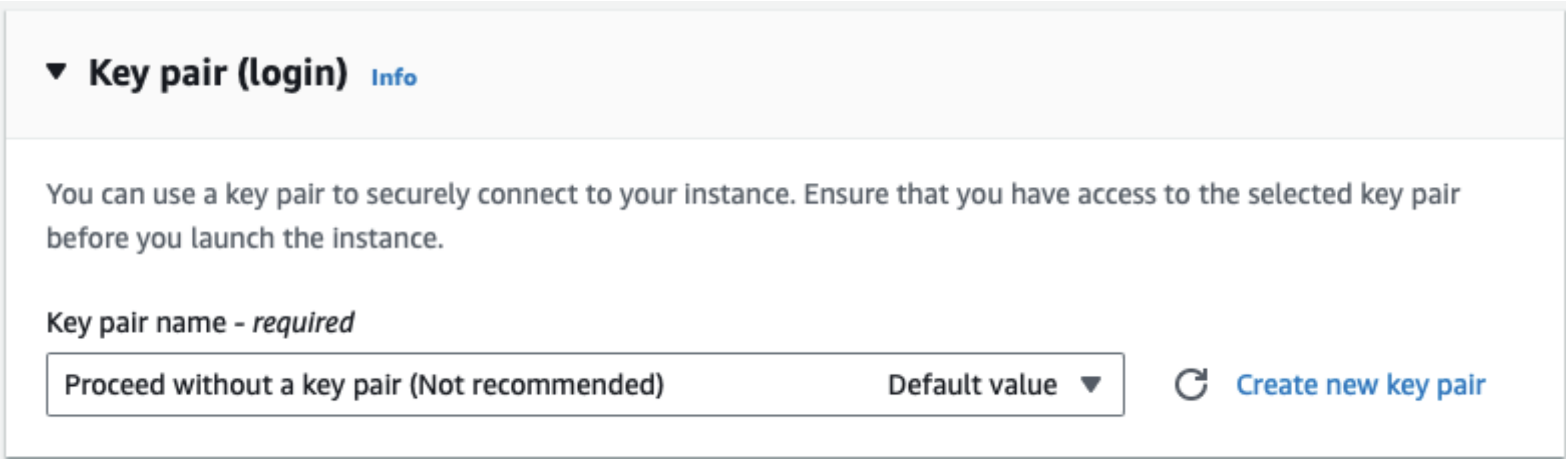


Instance type

z1d.2xlarge
Family: z1d   8 vCPU   64 GiB Memory   Current generation: true

The AMI vendor recommends using a z1d.2xlarge instance (or larger) for the best experience with this product.

Key pair



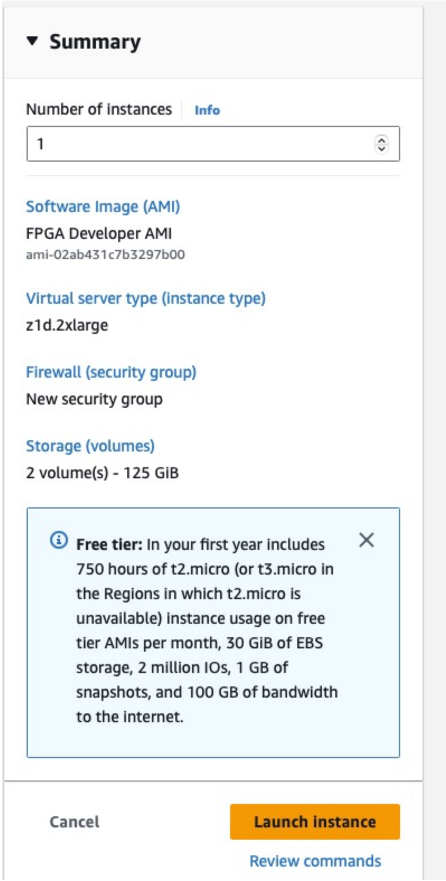
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Proceed without a key pair (Not recommended) Default value ▼

For information on how to create a new key pair go here

Launch z1d.2xlarge Instance



▼ Summary

Number of instances

1

Software Image (AMI)

FPGA Developer AMI

ami-02ab431c7b3297b00

Virtual server type (instance type)

z1d.2xlarge

Firewall (security group)

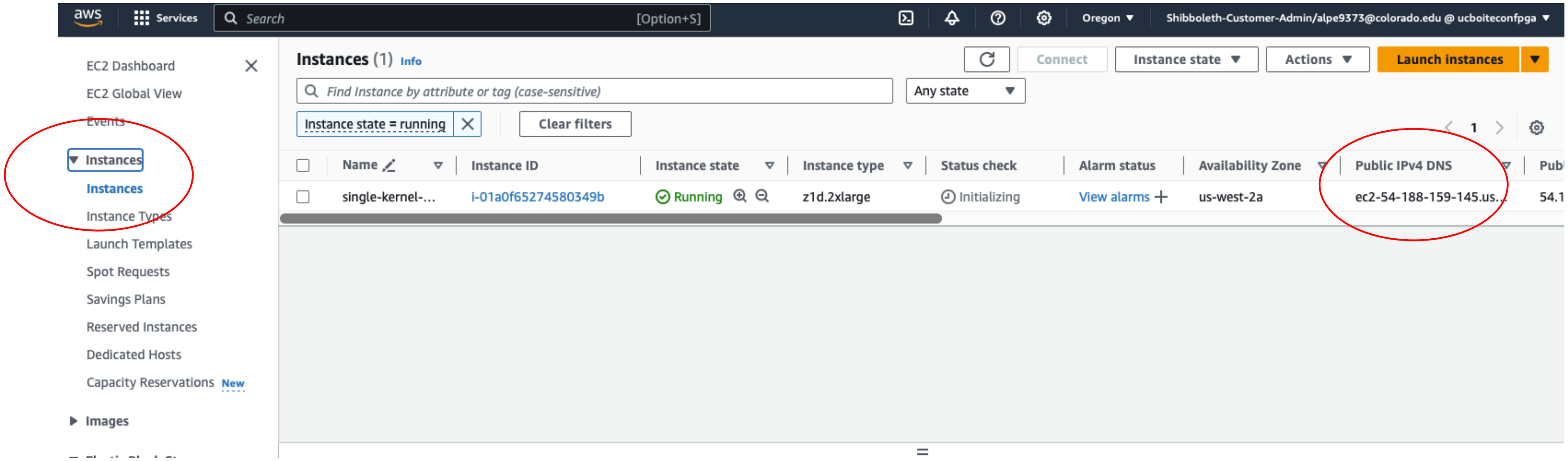
New security group

Storage (volumes)

2 volume(s) - 125 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, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

EC2 Instances

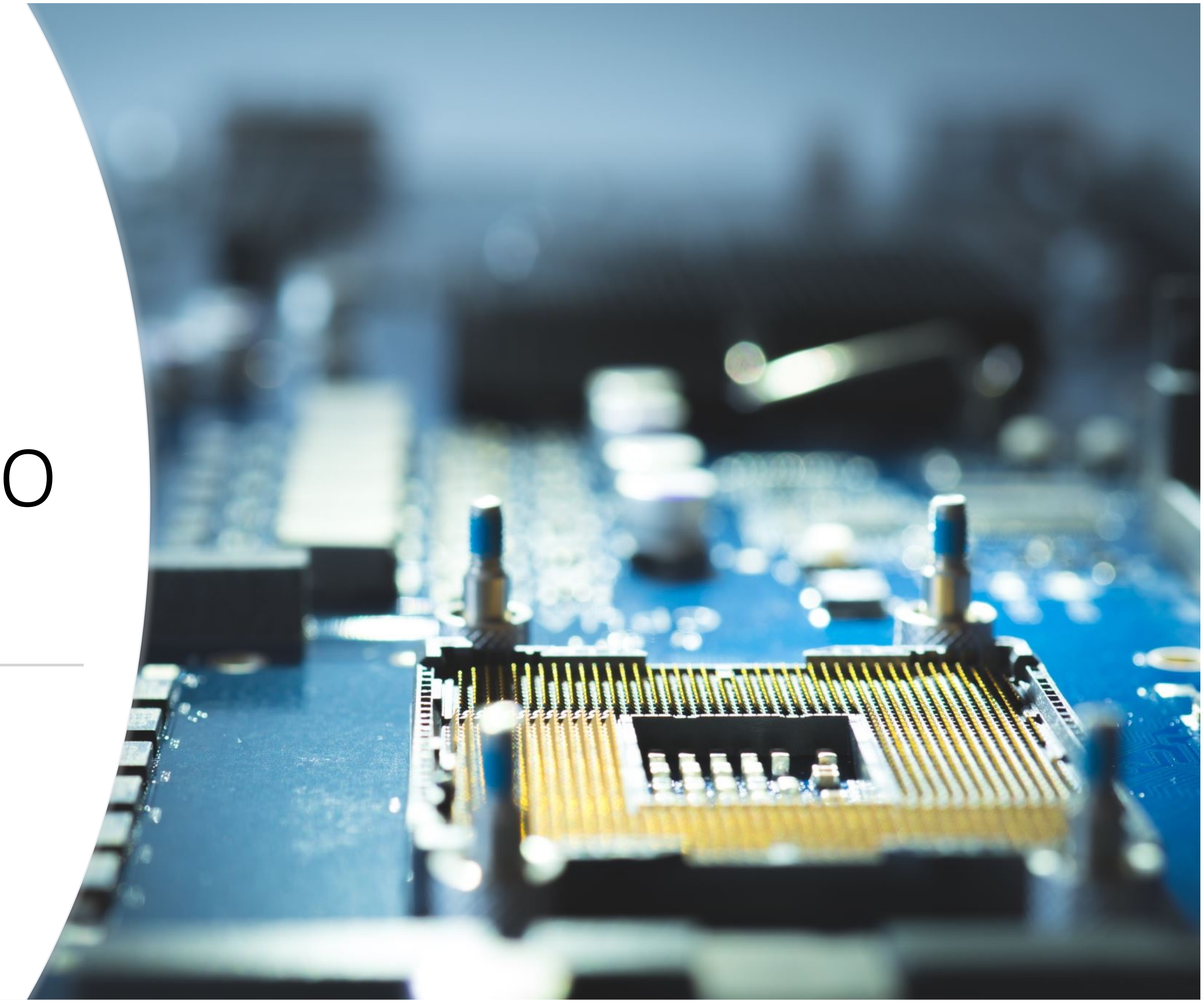


Instance state = running ✕

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Publ
□	single-kernel-...	i-01a0f65274580349b	✓ Running	z1d.2xlarge	Initializing	View alarms +	us-west-2a	ec2-54-188-159-145.us...	54.1

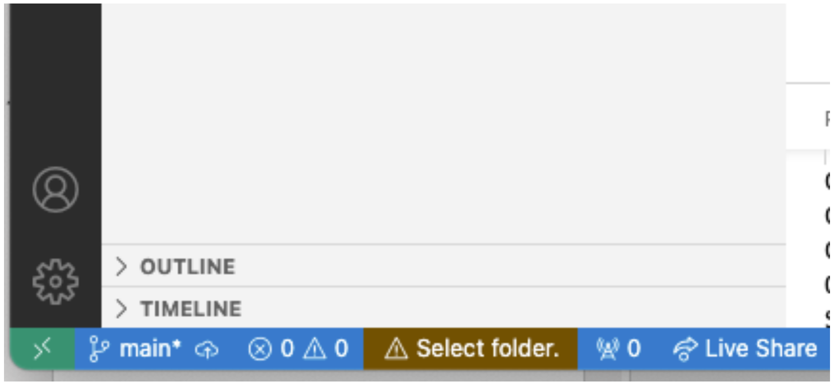
- In the top-left menu, select 'Instances'
- Copy the public IPv4 address in Visual Studio code

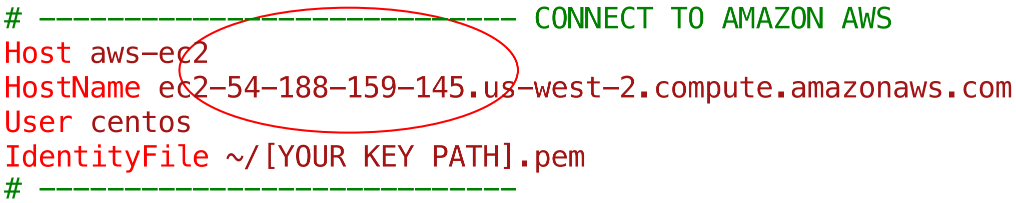
VISUAL STUDIO CODE



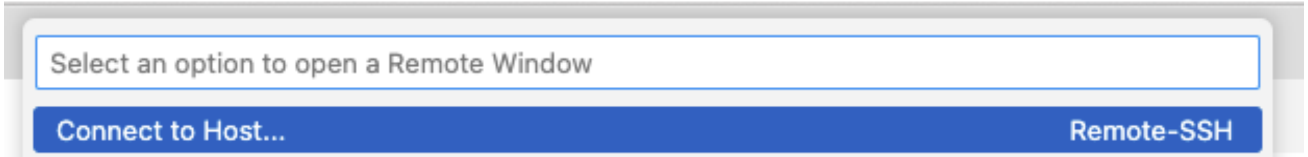
Open a Remote Window

- On the bottom-left corner of Visual Studio Code click on the green button ‘Open a Remote Window’
- Click on Connect to Host
- Click on Configure SSH Hosts
- Copy the public IP address
- Connect to aws-ec2.
- If you receive an error try to set: User root (in place of User Centos)





```
# --------------------------- CONNECT TO AMAZON AWS
Host aws-ec2
HostName ec2-54-188-159-145.us-west-2.compute.amazonaws.com
User centos
IdentityFile ~/[YOUR KEY PATH].pem
# ---------------------------
```



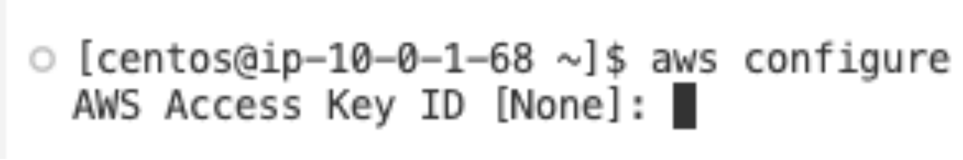
Setup the Instance

Clone the Github Repos

- Clone our GitHub repository into a directory of your preference (e.g., /home/centos):

```
git clone https://github.com/aws/aws-fpga.git $AWS_FPGA_REPO_DIR
git clone https://github.com/AleP83/FPGA-Econ.git
```

AWS Configure



```
○ [centos@ip-10-0-1-68 ~]$ aws configure
  AWS Access Key ID [None]: █
```

1. Go to your aws account and set (one time thing):
   - AWS Access Key ID
   -AWS Secret Access Key
2. Go to the terminal in visual studio and type aws configure

```
aws configure
 AWS Access Key ID [************xxxx]: <Your AWS Access Key ID>
 AWS Secret Access Key [*************xxxx]: <Your AWS Secret Access Key>
 Default region name: [us-west-1]: us-east-1
 Default output format [None]: json
```

3. Set:
   - AWS Access Key ID:
   - AWS Secret Access Key
   - Default Region name: us-west-2
     Note: this depends in which region you launched your instance.
   - Default output format [json]: json

Modify the Makefile

- Set the AWS S3 Bucket Name. Specify the S3 bucket name by replacing S3-NAME-GOES-HERE

```
S3_EXE_BUCKET_NAME := S3-NAME-GOES-HERE
```

Remark: The S3 bucket name must be globally unique within AWS. If an error occurs during bucket creation, it may be due to the name being already in use by another user.

- Select the AWS region of the S3 bucket (default is us-west-2):

```
AWS_REGION := us-west-2
```

Modify the following files

1. common/app.cpp

Line 10: Set the number of models

-
  ```
  #define N_MODEL 1200 // total number of models
  ```

2. common/definitions.h

Lines 41-42: Select the grid points

- /** Configure grid points.
- The analysis in the paper involves six distinct combinations: NKGRID={100,200,300}; NKM_GRID={4,8}.
- */
- #define NKGRID 100 // grid points on individual capital grid
- #define NKM_GRID 4 // grid points on aggregate capital grid

3. common/dev_options.h

```
// Select FPGA design by enabling exaclty one of the following macros
(setting it to one), keeping the rest to zero. For best performance, set
_ACROSS_ECONOMY to 1 and rest 0
#define _BASELINE 0 // FPGA design with no HLS acceleration.
#define _PIPELINE 0 // FPGA design with only PIPELINE acceleration
#define _WITHIN_ECONOMY 0 // FPGA design with one-kernel data
parallelization and pipelining
#define _ACROSS_ECONOMY 1 // FPGA design with three-kernels. Benchmark
```

5. fpga/design.cfg

- Select the kernel design: three-kernel design vs single-kernel design

Deafult: three-kernel design

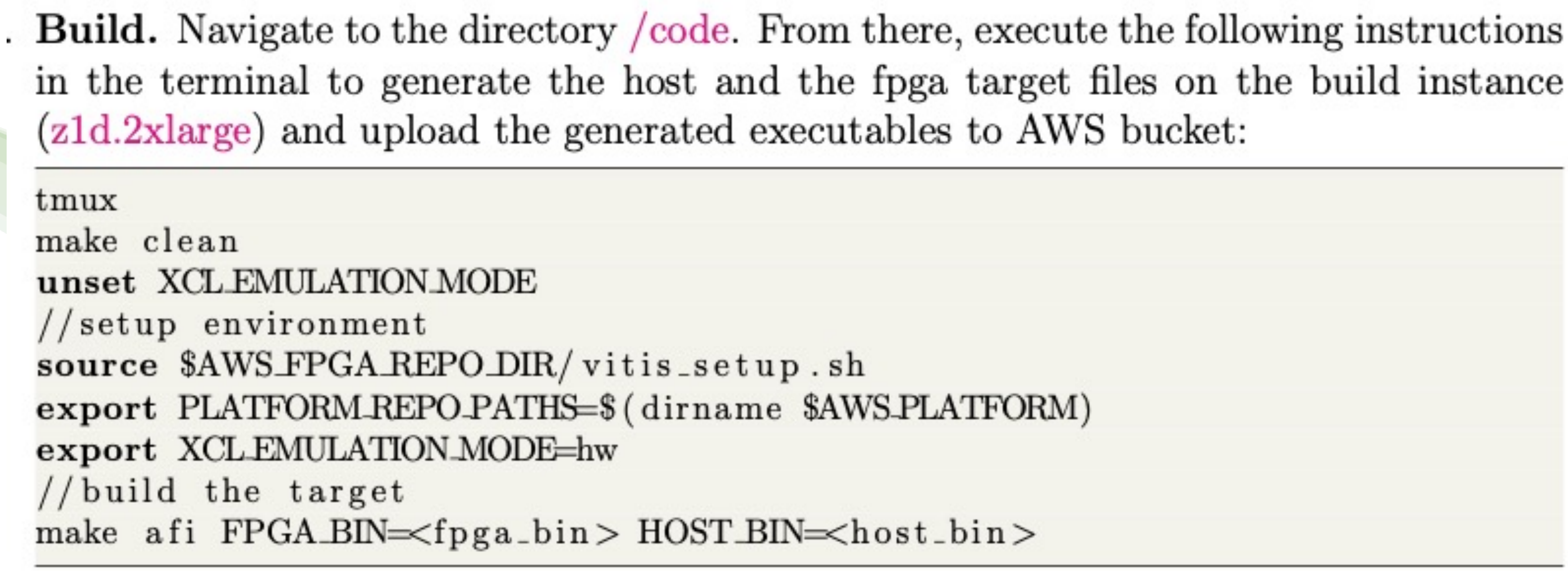
```
#Enable either single kernel or three kernel
##############single kernel start################
# [connectivity]
# nk=runOnfpga:1:runOnfpga_1
##############single kernel end##################
###############three kernel start################
[connectivity]
nk=runOnfpga:3:runOnfpga_1.runOnfpga_2.runOnfpga_3
```

Create the FPGA Images

In Visual Studio Code

- Open the terminal
- Go to the terminal

Follow instructions on readme from here



. **Build.** Navigate to the directory /code. From there, execute the following instructions in the terminal to generate the host and the fpga target files on the build instance (z1d.2xlarge) and upload the generated executables to AWS bucket:

```
tmux
make clean
unset XCL_EMULATION_MODE
//setup environment
source $AWS_FPGA_REPO_DIR/vitis_setup.sh
export PLATFORM_REPO_PATHS=$(dirname $AWS_PLATFORM)
export XCL_EMULATION_MODE=hw
//build the target
make afi FPGA_BIN=<fpga_bin> HOST_BIN=<host_bin>
```

Example: `make afi FPGA_BIN=3ker_100k_4km HOST_BIN=1200_3ker_100k_4km`

3-kernel-100-4 (1200 Economies)

/common/app.cpp

```
#define N_MODEL 1200 // total number of models
```

/common/definitions.h

```
#define NKGRID 100 // grid points on individual capital grid
#define NKM_GRID 4 // grid points on aggregate capital grid
```

/common/dev_options.h

```
// Select FPGA design by enabling exactly one of the following macros, keeping the rest to zero. For best performance,
set _ACROSS_ECONOMY to 1 and rest 0
#define _BASELINE 0 // FPGA design with no HLS acceleration.
#define _PIPELINE 0 // FPGA design with only PIPELINE acceleration
#define _WITHIN_ECONOMY 0 // FPGA design with one-kernel data parallelization and pipelining
#define _ACROSS_ECONOMY 1 // FPGA design with three-kernels. Benchmark
```

/fpga/design.cfg

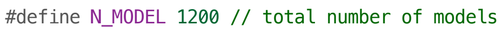
```
###############three kernel start################
[connectivity]
nk=runOnfpga:3:runOnfpga_1.runOnfpga_2.runOnfpga_3
slr=runOnfpga_1:SLR2
slr=runOnfpga_2:SLR1
slr=runOnfpga_3:SLR0
sp=runOnfpga_1.m_axi_gmem0:DDR[1]
sp=runOnfpga_2.m_axi_gmem0:DDR[0]
sp=runOnfpga_3.m_axi_gmem0:DDR[3]
###############three kernel end##################
[vivado]
prop=run.impl_1.strategy=Performance_ExtraTimingOpt
```

Terminal launch:

```
tmux
make clean
unset XCL_EMULATION_MODE
source $AWS_FPGA_REPO_DIR/vitis_setup.sh
export PLATFORM_REPO_PATHS=$(dirname $AWS_PLATFORM)
export XCL_EMULATION_MODE=hw
make afi FPGA_BIN=3ker_100k_4km HOST_BIN=1200_3ker_100k_4km
```

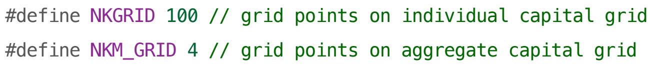
1-kernel-100-4 (1200 Economies)

/common/app.cpp



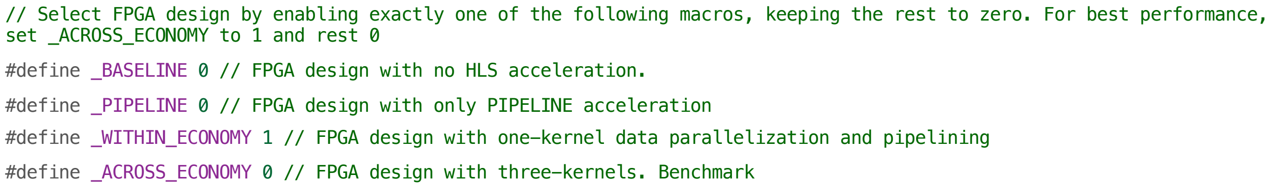
```
#define N_MODEL 1200 // total number of models
```

/common/definitions.h



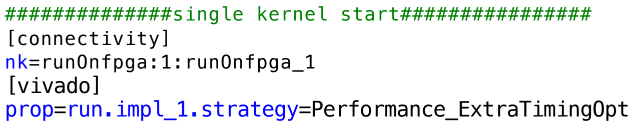
```
#define NKGRID 100 // grid points on individual capital grid
#define NKM_GRID 4 // grid points on aggregate capital grid
```

/common/dev_options.h



```
// Select FPGA design by enabling exactly one of the following macros, keeping the rest to zero. For best performance,
set _ACROSS_ECONOMY to 1 and rest 0
#define _BASELINE 0 // FPGA design with no HLS acceleration.
#define _PIPELINE 0 // FPGA design with only PIPELINE acceleration
#define _WITHIN_ECONOMY 1 // FPGA design with one-kernel data parallelization and pipelining
#define _ACROSS_ECONOMY 0 // FPGA design with three-kernels. Benchmark
```

/fpga/design.cfg



```
##############single kernel start################
[connectivity]
nk=runOnfpga:1:runOnfpga_1
[vivado]
prop=run.impl_1.strategy=Performance_ExtraTimingOpt
```

Terminal launch:

```
tmux
make clean
unset XCL_EMULATION_MODE
source $AWS_FPGA_REPO_DIR/vitis_setup.sh
export PLATFORM_REPO_PATHS=$(dirname $AWS_PLATFORM)
export XCL_EMULATION_MODE=hw
make afi FPGA_BIN=1ker_100k_4km HOST_BIN=1200_1ker_100k_4km
```

1-kernel-200-4 (1200 Economies)

/common/app.cpp

```
#define N_MODEL 1200 // total number of models
```

/common/definitions.h

```
#define NKGRID 200 // grid points on individual capital grid
#define NKM_GRID 4 // grid points on aggregate capital grid
```

/common/dev_options.h

```
// Select FPGA design by enabling exactly one of the following macros, keeping the rest to zero. For best performance,
set _ACROSS_ECONOMY to 1 and rest 0
#define _BASELINE 0 // FPGA design with no HLS acceleration.
#define _PIPELINE 0 // FPGA design with only PIPELINE acceleration
#define _WITHIN_ECONOMY 1 // FPGA design with one-kernel data parallelization and pipelining
#define _ACROSS_ECONOMY 0 // FPGA design with three-kernels. Benchmark
```

/fpga/design.cfg

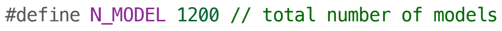
```
##############single kernel start################
[connectivity]
nk=runOnfpga:1:runOnfpga_1
[vivado]
prop=run.impl_1.strategy=Performance_ExtraTimingOpt
```

Terminal launch:

```
tmux
make clean
unset XCL_EMULATION_MODE
source $AWS_FPGA_REPO_DIR/vitis_setup.sh
export PLATFORM_REPO_PATHS=$(dirname $AWS_PLATFORM)
export XCL_EMULATION_MODE=hw
make afi FPGA_BIN=1ker_200k_4km HOST_BIN=1200_1ker_200k_4km
```

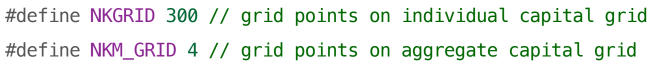
1-kernel-300-4 (1200 Economies)

/common/app.cpp



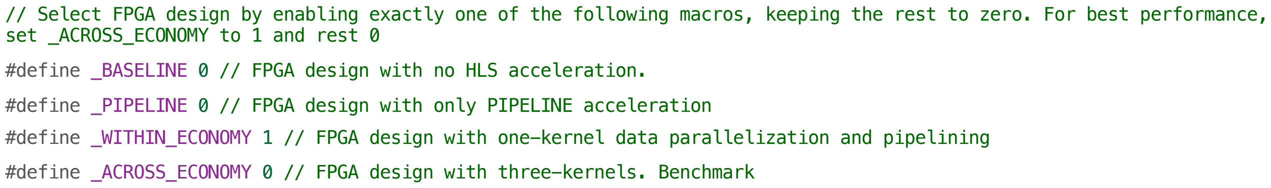
```
#define N_MODEL 1200 // total number of models
```

/common/definitions.h



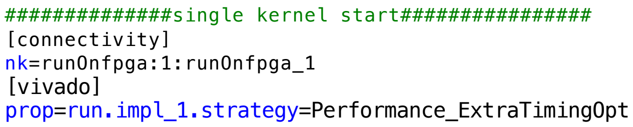
```
#define NKGRID 300 // grid points on individual capital grid
#define NKM_GRID 4 // grid points on aggregate capital grid
```

/common/dev_options.h



```
// Select FPGA design by enabling exactly one of the following macros, keeping the rest to zero. For best performance,
set _ACROSS_ECONOMY to 1 and rest 0
#define _BASELINE 0 // FPGA design with no HLS acceleration.
#define _PIPELINE 0 // FPGA design with only PIPELINE acceleration
#define _WITHIN_ECONOMY 1 // FPGA design with one-kernel data parallelization and pipelining
#define _ACROSS_ECONOMY 0 // FPGA design with three-kernels. Benchmark
```

/fpga/design.cfg



```
##############single kernel start################
[connectivity]
nk=runOnfpga:1:runOnfpga_1
[vivado]
prop=run.impl_1.strategy=Performance_ExtraTimingOpt
```

Terminal launch:

```
tmux
make clean
unset XCL_EMULATION_MODE
source $AWS_FPGA_REPO_DIR/vitis_setup.sh
export PLATFORM_REPO_PATHS=$(dirname $AWS_PLATFORM)
export XCL_EMULATION_MODE=hw
make afi FPGA_BIN=1ker_300k_4km HOST_BIN=1200_1ker_300k_4km
```

1-kernel-100-8 (1200 Economies)

/common/app.cpp

```
#define N_MODEL 1200 // total number of models
```

/common/definitions.h

```
#define NKGRID 100 // grid points on individual capital grid
#define NKM_GRID 8 // grid points on aggregate capital grid
```

/common/dev_options.h

```
// Select FPGA design by enabling exactly one of the following macros, keeping the rest to zero. For best performance,
set _ACROSS_ECONOMY to 1 and rest 0
#define _BASELINE 0 // FPGA design with no HLS acceleration.
#define _PIPELINE 0 // FPGA design with only PIPELINE acceleration
#define _WITHIN_ECONOMY 1 // FPGA design with one-kernel data parallelization and pipelining
#define _ACROSS_ECONOMY 0 // FPGA design with three-kernels. Benchmark
```

/fpga/design.cfg

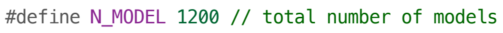
```
##############single kernel start################
[connectivity]
nk=runOnfpga:1:runOnfpga_1
[vivado]
prop=run.impl_1.strategy=Performance_ExtraTimingOpt
```

Terminal launch:

```
tmux
make clean
unset XCL_EMULATION_MODE
source $AWS_FPGA_REPO_DIR/vitis_setup.sh
export PLATFORM_REPO_PATHS=$(dirname $AWS_PLATFORM)
export XCL_EMULATION_MODE=hw
make afi FPGA_BIN=1ker_100k_8km HOST_BIN=1200_1ker_100k_8km
```

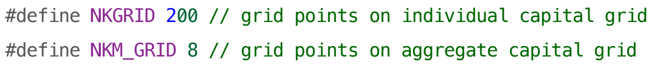
1-kernel-200-8 (1200 Economies)

/common/app.cpp



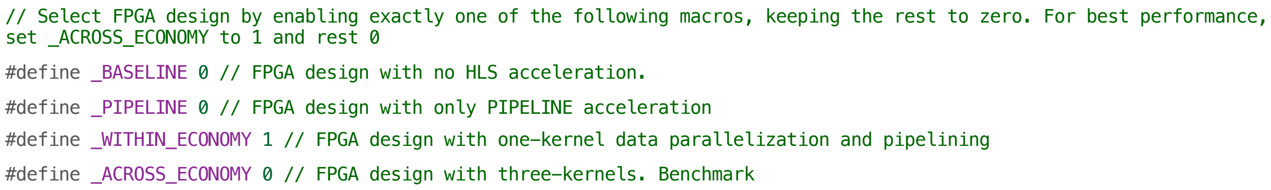
```
#define N_MODEL 1200 // total number of models
```

/common/definitions.h



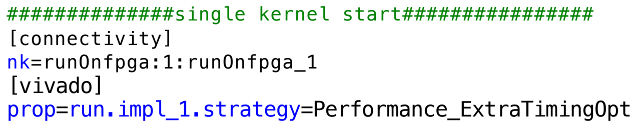
```
#define NKGRID 200 // grid points on individual capital grid
#define NKM_GRID 8 // grid points on aggregate capital grid
```

/common/dev_options.h



```
// Select FPGA design by enabling exactly one of the following macros, keeping the rest to zero. For best performance,
set _ACROSS_ECONOMY to 1 and rest 0
#define _BASELINE 0 // FPGA design with no HLS acceleration.
#define _PIPELINE 0 // FPGA design with only PIPELINE acceleration
#define _WITHIN_ECONOMY 1 // FPGA design with one-kernel data parallelization and pipelining
#define _ACROSS_ECONOMY 0 // FPGA design with three-kernels. Benchmark
```

/fpga/design.cfg



```
##############single kernel start################
[connectivity]
nk=runOnfpga:1:runOnfpga_1
[vivado]
prop=run.impl_1.strategy=Performance_ExtraTimingOpt
```

Terminal launch:

```
tmux
make clean
unset XCL_EMULATION_MODE
source $AWS_FPGA_REPO_DIR/vitis_setup.sh
export PLATFORM_REPO_PATHS=$(dirname $AWS_PLATFORM)
export XCL_EMULATION_MODE=hw
make afi FPGA_BIN=1ker_200k_8km HOST_BIN=1200_1ker_200k_8km
```

1-kernel-300-8 (1200 Economies)

/common/app.cpp

```
#define N_MODEL 1200 // total number of models
```

/common/definitions.h

```
#define NKGRID 300 // grid points on individual capital grid
#define NKM_GRID 8 // grid points on aggregate capital grid
```

/common/dev_options.h

```
// Select FPGA design by enabling exactly one of the following macros, keeping the rest to zero. For best performance,
set _ACROSS_ECONOMY to 1 and rest 0
#define _BASELINE 0 // FPGA design with no HLS acceleration.
#define _PIPELINE 0 // FPGA design with only PIPELINE acceleration
#define _WITHIN_ECONOMY 1 // FPGA design with one-kernel data parallelization and pipelining
#define _ACROSS_ECONOMY 0 // FPGA design with three-kernels. Benchmark
```

/fpga/design.cfg

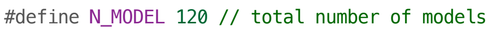
```
##############single kernel start################
[connectivity]
nk=runOnfpga:1:runOnfpga_1
[vivado]
prop=run.impl_1.strategy=Performance_ExtraTimingOpt
```

Terminal launch:

```
tmux
make clean
unset XCL_EMULATION_MODE
source $AWS_FPGA_REPO_DIR/vitis_setup.sh
export PLATFORM_REPO_PATHS=$(dirname $AWS_PLATFORM)
export XCL_EMULATION_MODE=hw
make afi FPGA_BIN=1ker_300k_8km HOST_BIN=1200_1ker_300k_8km
```

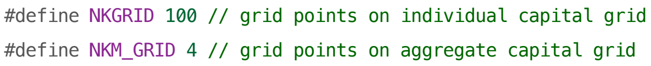
1-kernel-100-4-Baseline (120 Economies)

/common/app.cpp



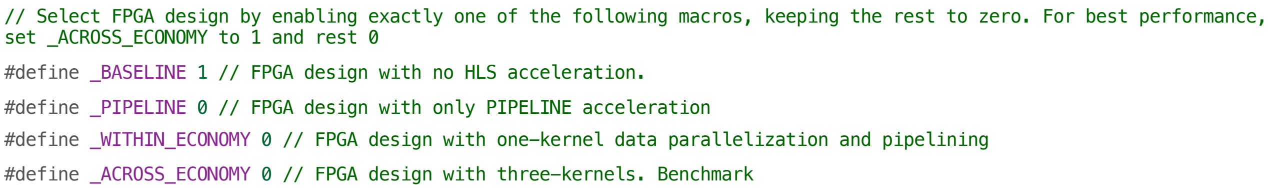
```
#define N_MODEL 120 // total number of models
```

/common/definitions.h



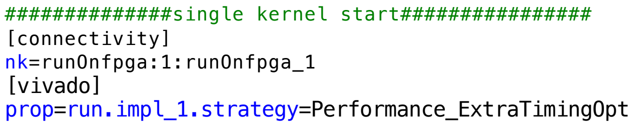
```
#define NKGRID 100 // grid points on individual capital grid
#define NKM_GRID 4 // grid points on aggregate capital grid
```

/common/dev_options.h



```
// Select FPGA design by enabling exactly one of the following macros, keeping the rest to zero. For best performance,
set _ACROSS_ECONOMY to 1 and rest 0
#define _BASELINE 1 // FPGA design with no HLS acceleration.
#define _PIPELINE 0 // FPGA design with only PIPELINE acceleration
#define _WITHIN_ECONOMY 0 // FPGA design with one-kernel data parallelization and pipelining
#define _ACROSS_ECONOMY 0 // FPGA design with three-kernels. Benchmark
```

/fpga/design.cfg



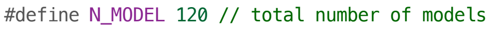
```
##############single kernel start################
[connectivity]
nk=runOnfpga:1:runOnfpga_1
[vivado]
prop=run.impl_1.strategy=Performance_ExtraTimingOpt
```

Terminal launch:

```
tmux
make clean
unset XCL_EMULATION_MODE
source $AWS_FPGA_REPO_DIR/vitis_setup.sh
export PLATFORM_REPO_PATHS=$(dirname $AWS_PLATFORM)
export XCL_EMULATION_MODE=hw
make afi FPGA_BIN=baseline_1ker_100k_4km HOST_BIN=120_1ker_100k_4km
```

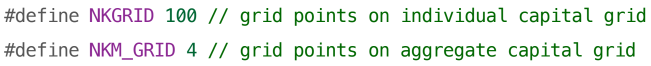
1-kernel-100-4-Pipeline (120 Economies)

/common/app.cpp



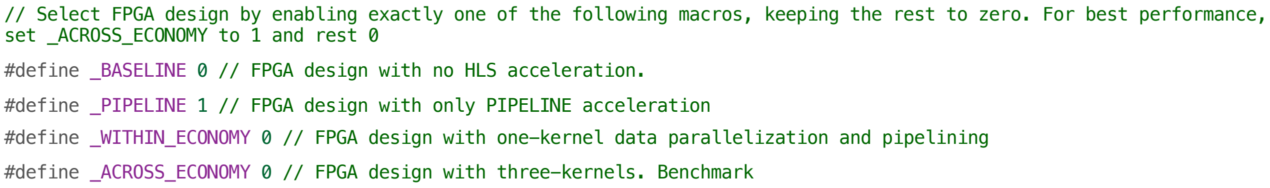
```
#define N_MODEL 120 // total number of models
```

/common/definitions.h



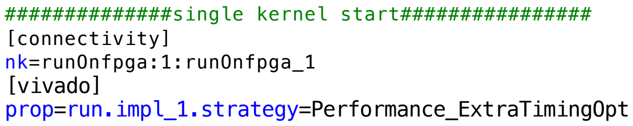
```
#define NKGRID 100 // grid points on individual capital grid
#define NKM_GRID 4 // grid points on aggregate capital grid
```

/common/dev_options.h



```
// Select FPGA design by enabling exactly one of the following macros, keeping the rest to zero. For best performance,
set _ACROSS_ECONOMY to 1 and rest 0
#define _BASELINE 0 // FPGA design with no HLS acceleration.
#define _PIPELINE 1 // FPGA design with only PIPELINE acceleration
#define _WITHIN_ECONOMY 0 // FPGA design with one-kernel data parallelization and pipelining
#define _ACROSS_ECONOMY 0 // FPGA design with three-kernels. Benchmark
```

/fpga/design.cfg



```
##############single kernel start################
[connectivity]
nk=runOnfpga:1:runOnfpga_1
[vivado]
prop=run.impl_1.strategy=Performance_ExtraTimingOpt
```

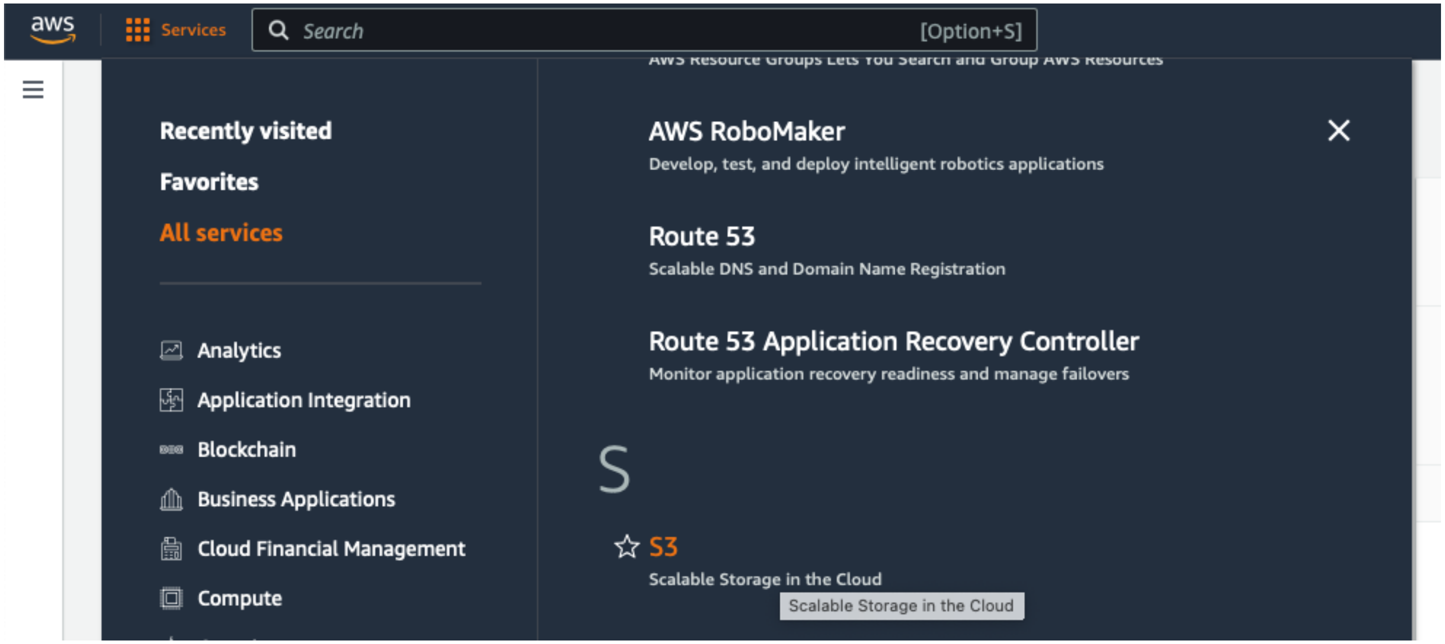
Terminal launch:

```
tmux
make clean
unset XCL_EMULATION_MODE
source $AWS_FPGA_REPO_DIR/vitis_setup.sh
export PLATFORM_REPO_PATHS=$(dirname $AWS_PLATFORM)
export XCL_EMULATION_MODE=hw
make afi FPGA_BIN=pipeline_1ker_100k_4km HOST_BIN=120_1ker_100k_4km
```

Delete S3 Buckets

S3 Bucket

- Log in your AWS Account
- Navigate > Services > All Services > S3



AWS Resource Groups Lets You Search and Group AWS Resources

AWS RoboMaker
Develop, test, and deploy intelligent robotics applications

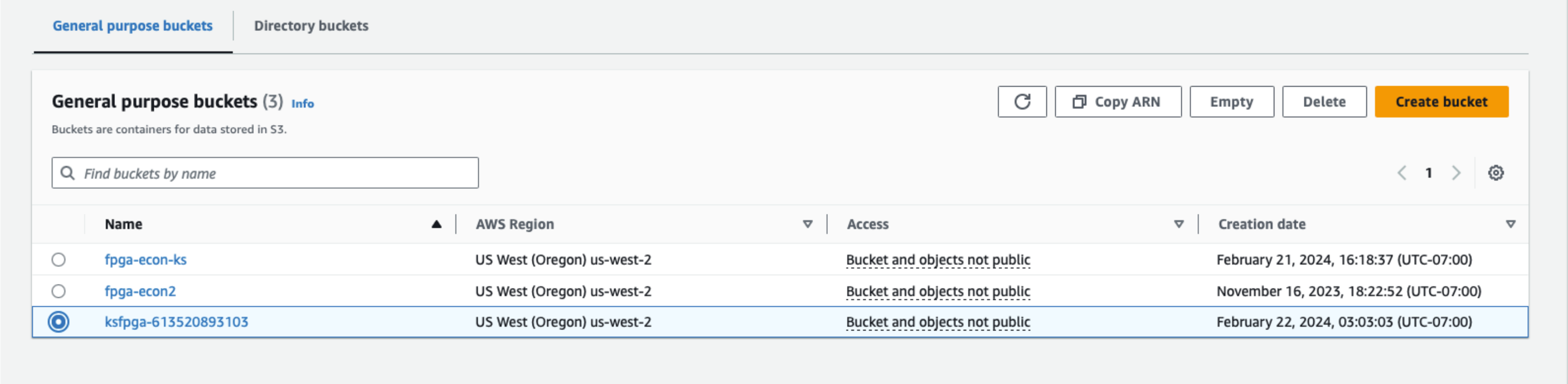
Route 53
Scalable DNS and Domain Name Registration

Route 53 Application Recovery Controller
Monitor application recovery readiness and manage failovers

☆ S3
Scalable Storage in the Cloud
Scalable Storage in the Cloud

Delete the Temporary bucket

- In this example the temporary bucket is named: ksfpga-613520893103
- First empty the bucket and then delete it
- Follow instructions here



General purpose buckets (3)

Buckets are containers for data stored in S3.

	Name	AWS Region	Access	Creation date
○	fpga-econ-ks	US West (Oregon) us-west-2	Bucket and objects not public	February 21, 2024, 16:18:37 (UTC-07:00)
○	fpga-econ2	US West (Oregon) us-west-2	Bucket and objects not public	November 16, 2023, 18:22:52 (UTC-07:00)
●	ksfpga-613520893103	US West (Oregon) us-west-2	Bucket and objects not public	February 22, 2024, 03:03:03 (UTC-07:00)