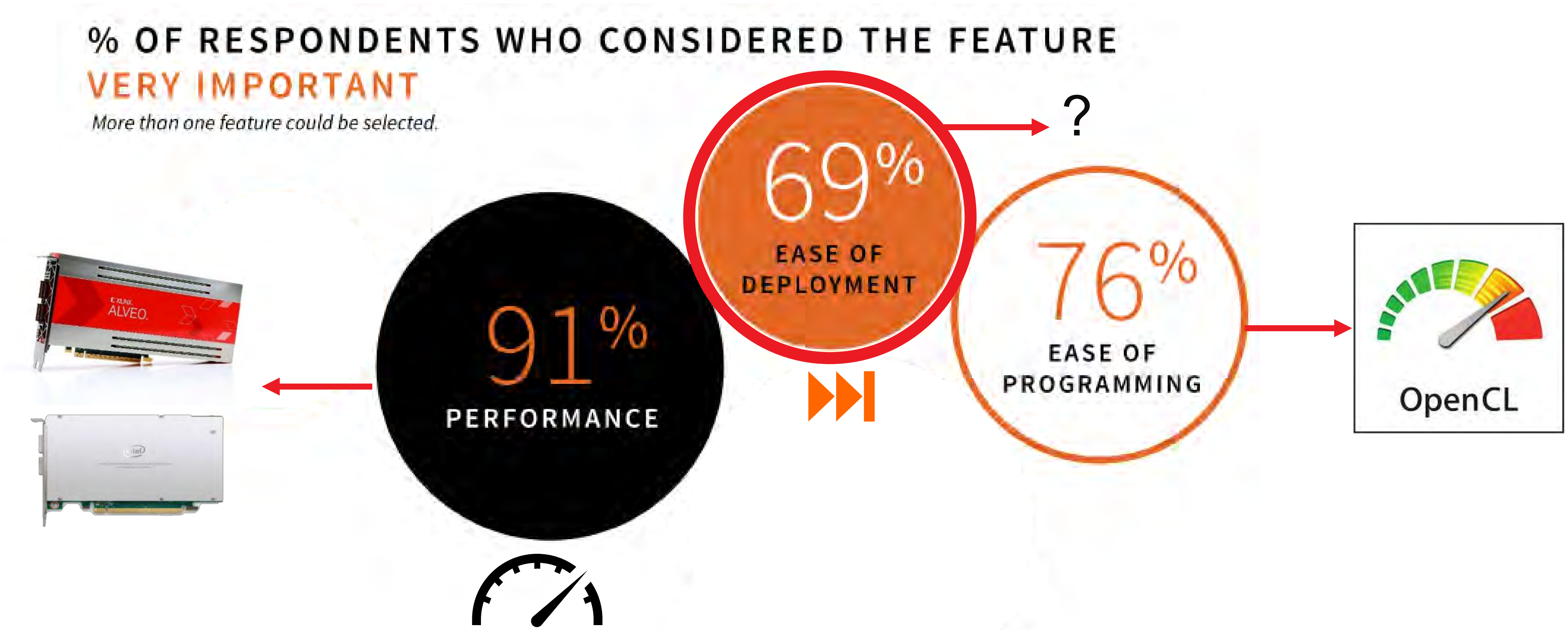
inaccel

Application Acceleration made simple

www.inaccel.com

info@inaccel.com

What software developers want



% OF RESPONDENTS WHO CONSIDERED THE FEATURE VERY IMPORTANT

More than one feature could be selected.

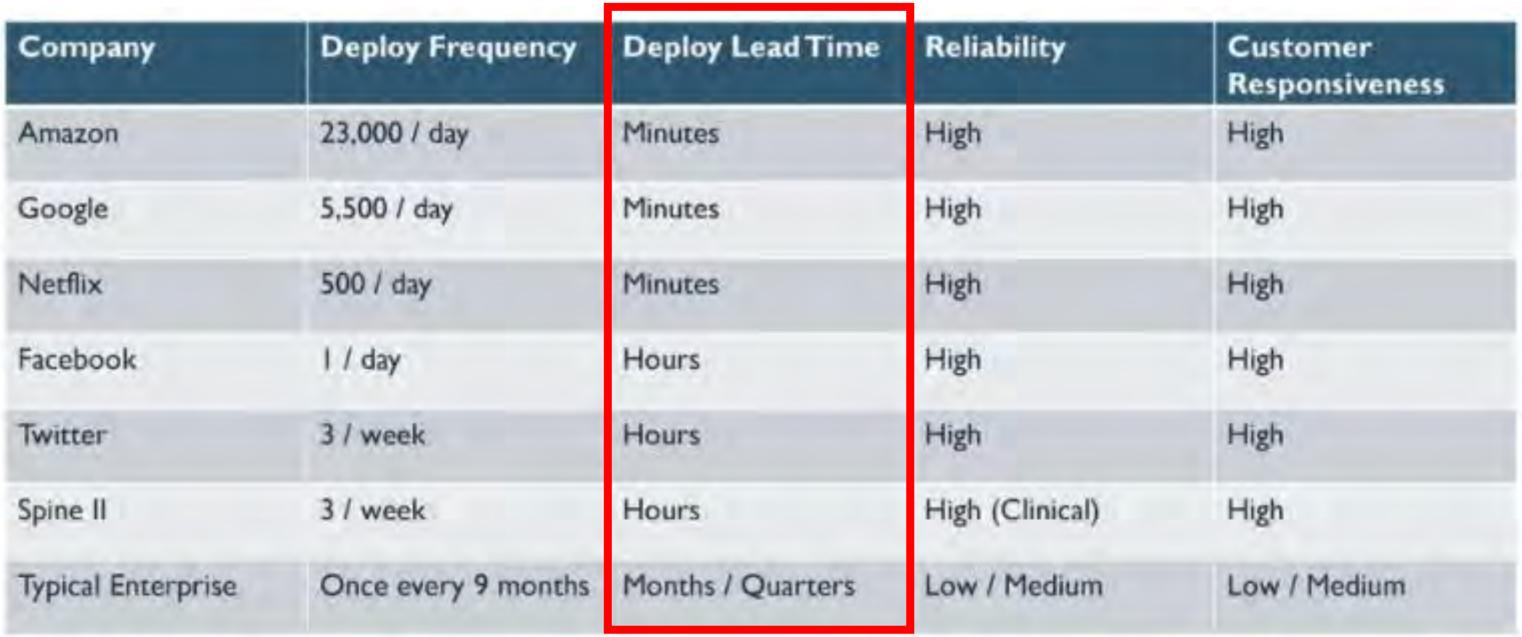
91% PERFORMANCE

69% EASE OF DEPLOYMENT

76% EASE OF PROGRAMMING

Source: Databricks, Apache Spark Survey 2016, Report

DevOps using CPUs, GPUs

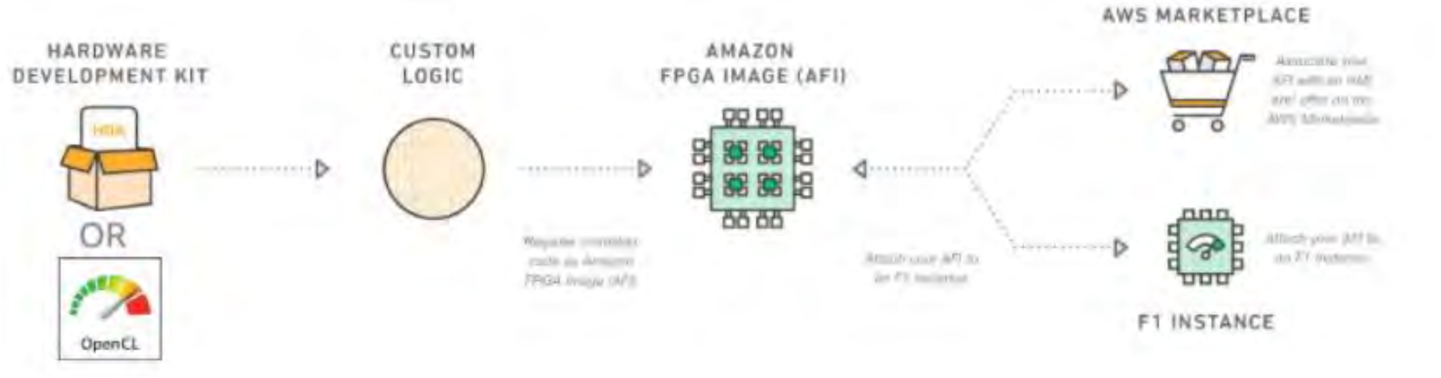


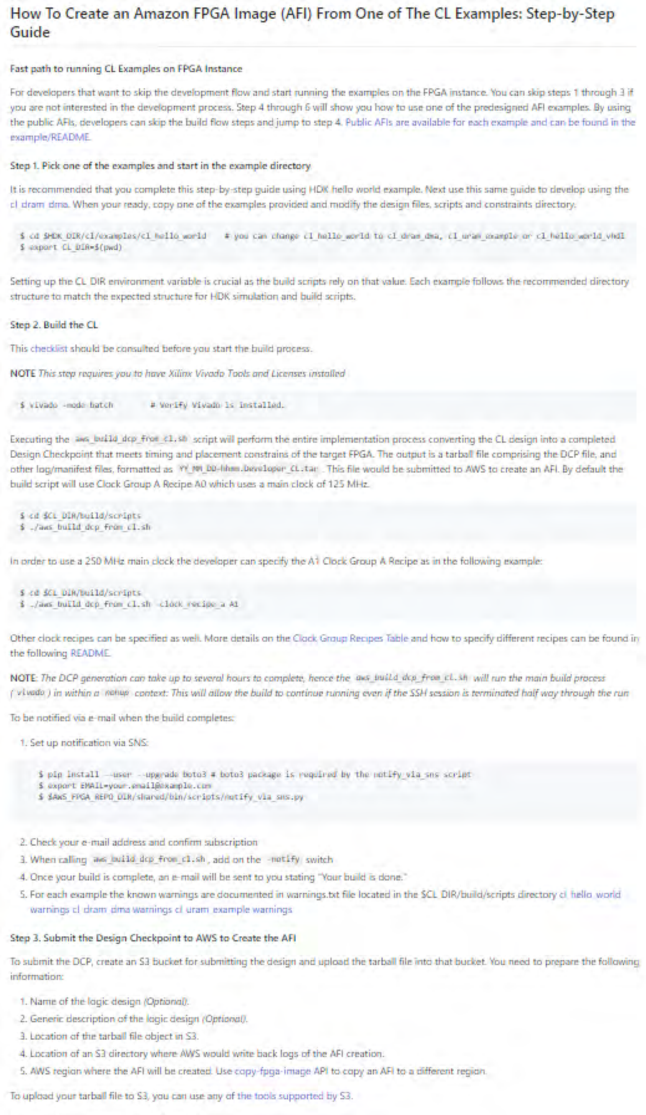
Company	Deploy Frequency	Deploy Lead Time	Reliability	Customer Responsiveness
Amazon	23,000 / day	Minutes	High	High
Google	5,500 / day	Minutes	High	High
Netflix	500 / day	Minutes	High	High
Facebook	1 / day	Hours	High	High
Twitter	3 / week	Hours	High	High
Spine II	3 / week	Hours	High (Clinical)	High
Typical Enterprise	Once every 9 months	Months / Quarters	Low / Medium	Low / Medium

Source: The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win is the third book by Gene Kim

Deploy FPGAs on cloud

- **Several steps**
- **Prior knowledge on FPGAs**
  - Bitstream
  - Memory management
  - Communication
  - Challenges: Bitstream version, Firmware, SDK


HARDWARE DEVELOPMENT KIT
HDK
OR
OpenCL
CUSTOM LOGIC
Register complete code as Amazon FPGA Image (AFI)
AMAZON FPGA IMAGE (AFI)
Attach your AFI to an F1 instance
AWS MARKETPLACE
Associate your AFI with an AMI and offer on the AWS Marketplace
F1 INSTANCE
Attach your AFI to an F1 instance



How To Create an Amazon FPGA Image (AFI) From One of The CL Examples: Step-by-Step Guide

Fast path to running CL Examples on FPGA Instance

For developers that want to skip the development flow and start running the examples on the FPGA instance. You can skip steps 1 through 3 if you are not interested in the development process. Step 4 through 6 will show you how to use one of the predesigned AFI examples. By using the public AFIs, developers can skip the build flow steps and jump to step 4. Public AFIs are available for each example and can be found in the example/README.

Step 1. Pick one of the examples and start in the example directory

It is recommended that you complete this step-by-step guide using HDK hello world example. Next use this same guide to develop using the cl dram dma. When your ready, copy one of the examples provided and modify the design files, scripts and constraints directory.

```
$ cd $HDK_DIR/cl/examples/cl_hello_world    # you can change cl_hello_world to cl_dram_dma, cl_uram_example or cl_hello_world_vhdl
$ export CL_DIR=$(pwd)
```

Setting up the CL DIR environment variable is crucial as the build scripts rely on that value. Each example follows the recommended directory structure to match the expected structure for HDK simulation and build scripts.

Step 2. Build the CL

This checklist should be consulted before you start the build process.

NOTE *This step requires you to have Xilinx Vivado Tools and Licenses installed*

```
$ vivado -mode batch       # Verify Vivado is installed.
```

Executing the `aws_build_dcp_from_cl.sh` script will perform the entire implementation process converting the CL design into a completed Design Checkpoint that meets timing and placement constrains of the target FPGA. The output is a tarball file comprising the DCP file, and other log/manifest files, formatted as `YY_MM_DD-hhmm.Developer_CL.tar`. This file would be submitted to AWS to create an AFI. By default the build script will use Clock Group A Recipe A0 which uses a main clock of 125 MHz.

```
$ cd $CL_DIR/build/scripts
$ ./aws_build_dcp_from_cl.sh
```

In order to use a 250 MHz main clock the developer can specify the A1 Clock Group A Recipe as in the following example:

```
$ cd $CL_DIR/build/scripts
$ ./aws_build_dcp_from_cl.sh -clock_recipe_a A1
```

Other clock recipes can be specified as well. More details on the Clock Group Recipes Table and how to specify different recipes can be found in the following README.

NOTE: *The DCP generation can take up to several hours to complete, hence the* `aws_build_dcp_from_cl.sh` *will run the main build process (* `vivado` *) in within a* `nohup` *context: This will allow the build to continue running even if the SSH session is terminated half way through the run*

To be notified via e-mail when the build completes:

1. Set up notification via SNS:

   ```
   $ pip install --user --upgrade boto3 # boto3 package is required by the notify_via_sns script
   $ export EMAIL=your.email@example.com
   $ $AWS_FPGA_REPO_DIR/shared/bin/scripts/notify_via_sns.py
   ```

2. Check your e-mail address and confirm subscription
3. When calling `aws_build_dcp_from_cl.sh`, add on the `-notify` switch
4. Once your build is complete, an e-mail will be sent to you stating "Your build is done."
5. For each example the known warnings are documented in warnings.txt file located in the $CL DIR/build/scripts directory cl hello world warnings cl dram dma warnings cl uram example warnings

Step 3. Submit the Design Checkpoint to AWS to Create the AFI

To submit the DCP, create an S3 bucket for submitting the design and upload the tarball file into that bucket. You need to prepare the following information:

1. Name of the logic design *(Optional)*.
2. Generic description of the logic design *(Optional)*.
3. Location of the tarball file object in S3.
4. Location of an S3 directory where AWS would write back logs of the AFI creation.
5. AWS region where the AFI will be created. Use copy-fpga-image API to copy an AFI to a different region.

To upload your tarball file to S3, you can use any of the tools supported by S3.

Challenges on FPGAs

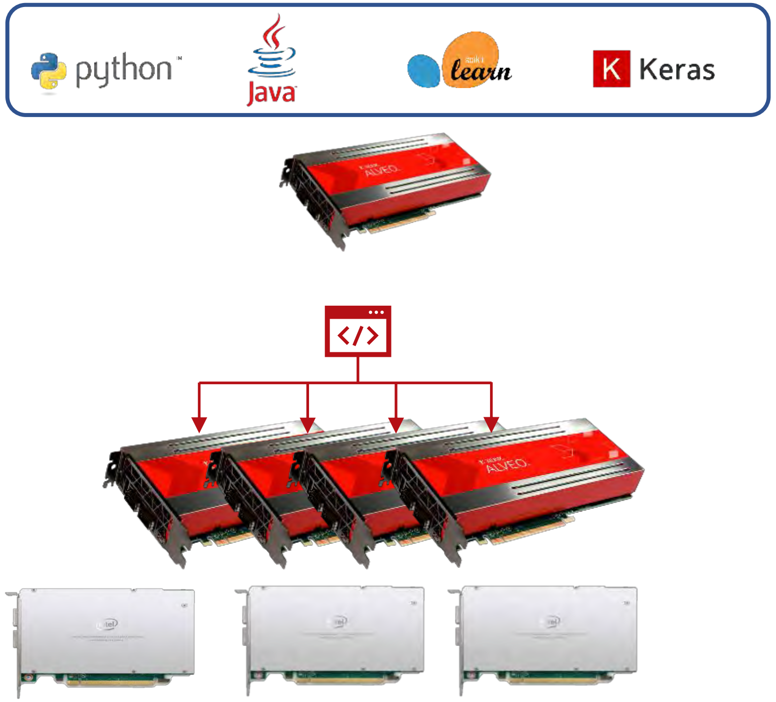
- > How can I deploy my FPGA accelerator easy?

python
Java
learn
Keras

Challenges

- > **How can I deploy my FPGA accelerator easy?**

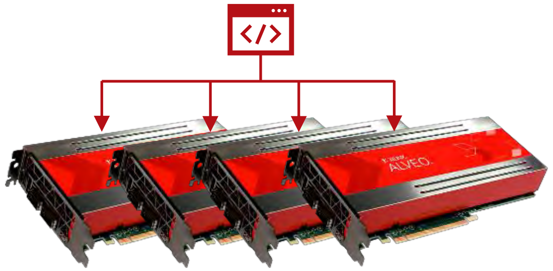
- > **How can I scale-out my applications to multiple FPGAs?**



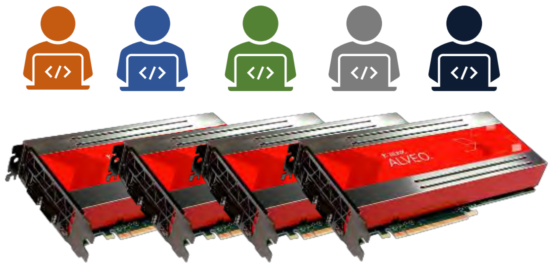
Challenges

- > **How can I deploy my FPGA accelerator easy?**

- > **How can I scale-out my applications to multiple Alveo cards?**

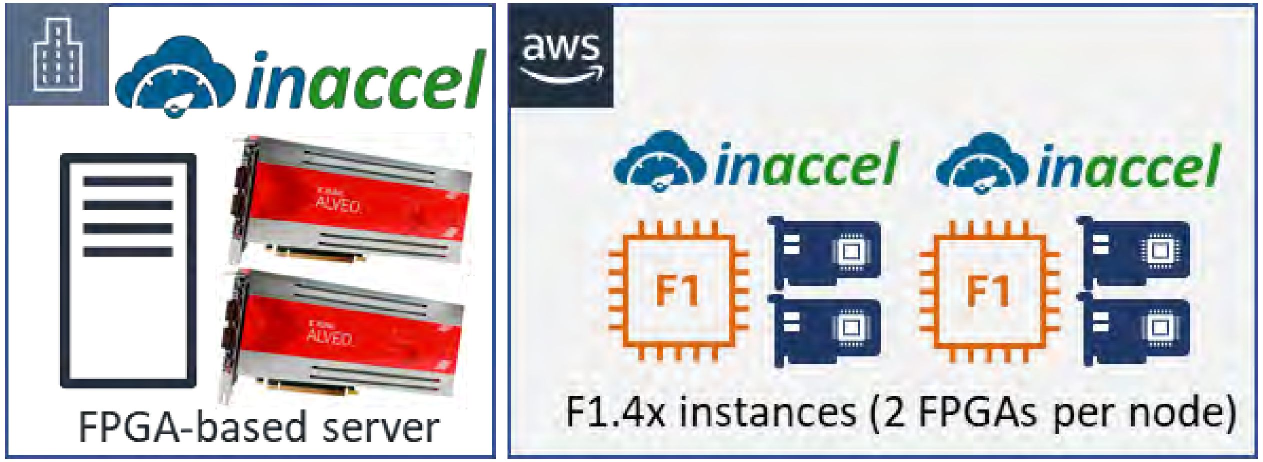


- > **How multiple users or applications can share my FPGA cluster?**

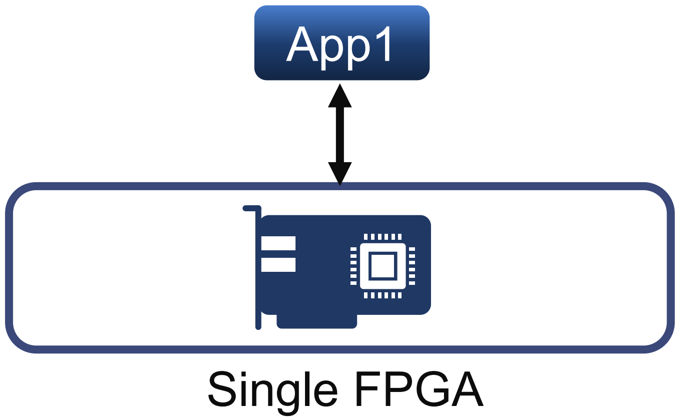


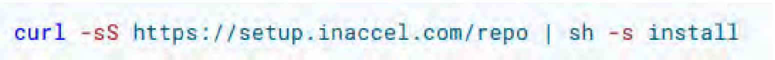
More Challenges

> **How can scale-out my application on-prem and on cloud?**

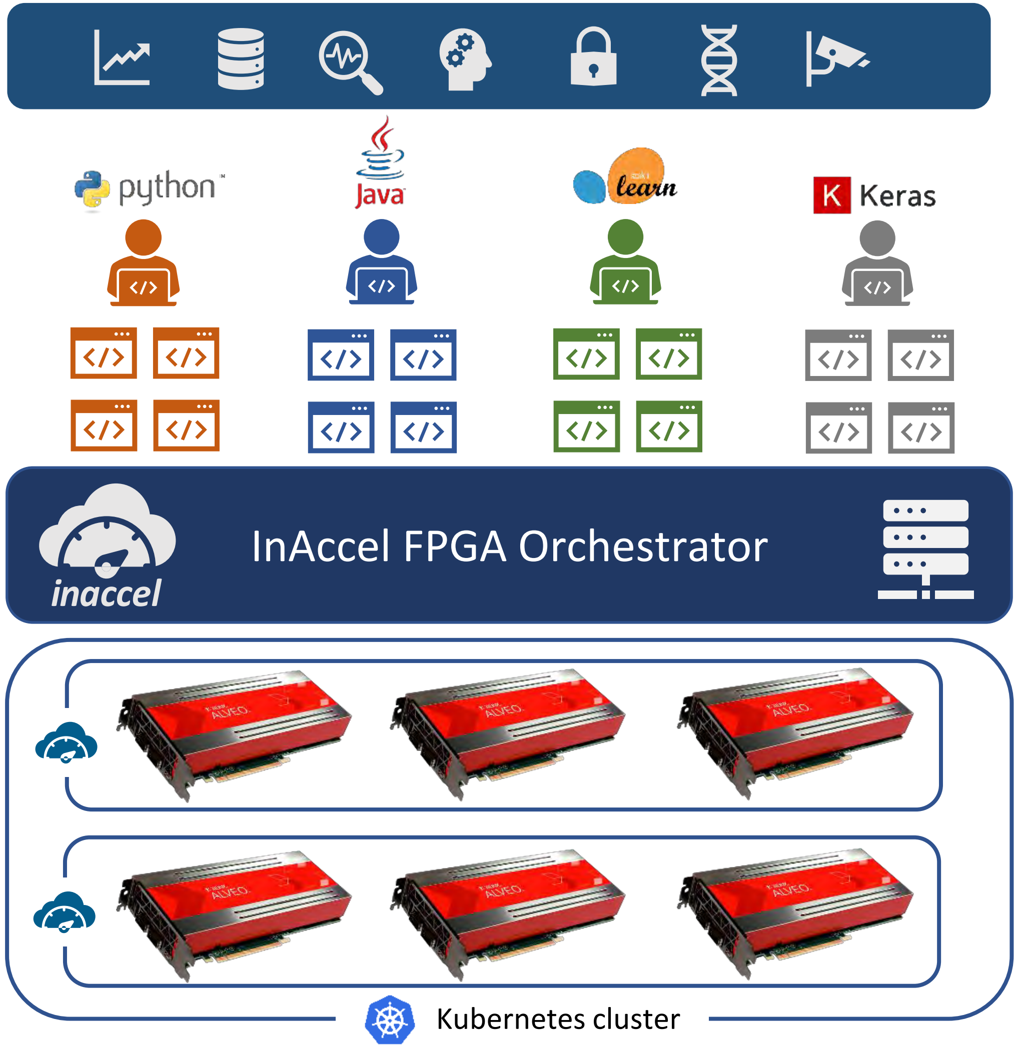


From single node to scalable deployment

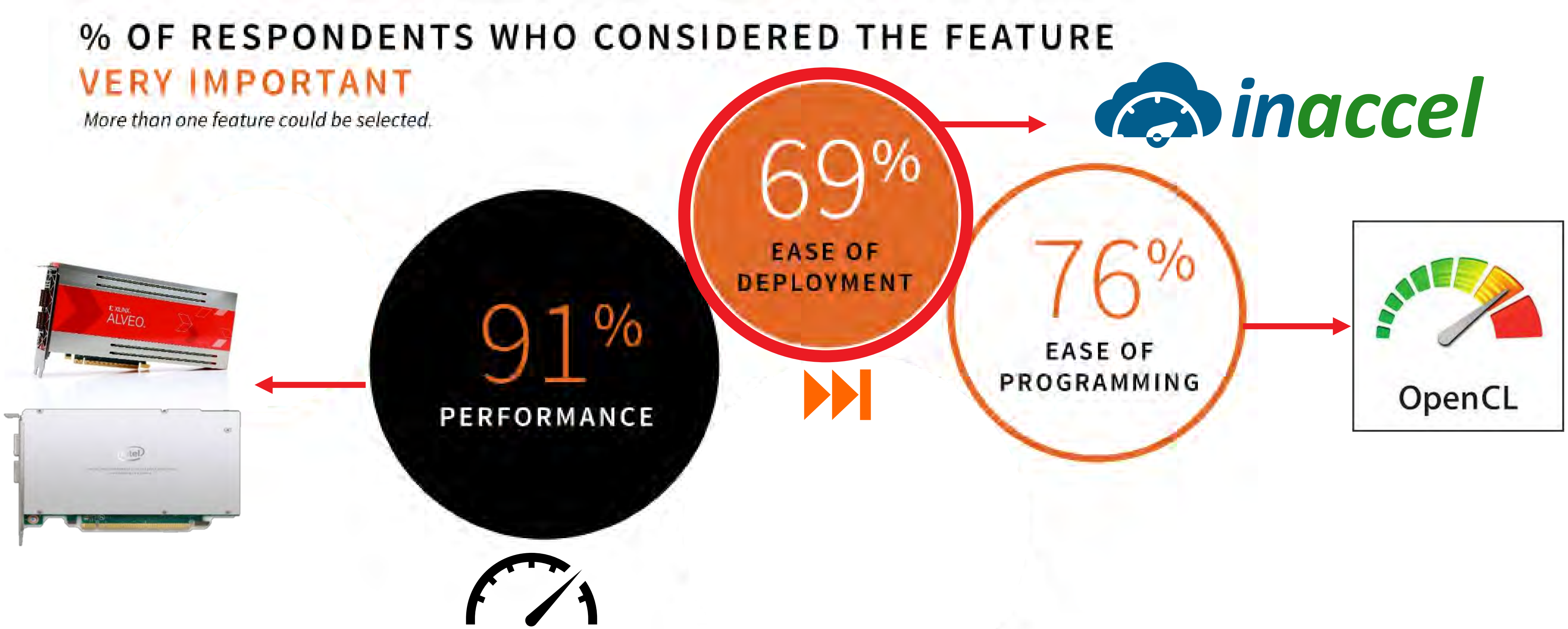




```
curl -sS https://setup.inaccel.com/repo | sh -s install
```



What software developers want



% OF RESPONDENTS WHO CONSIDERED THE FEATURE VERY IMPORTANT

More than one feature could be selected.

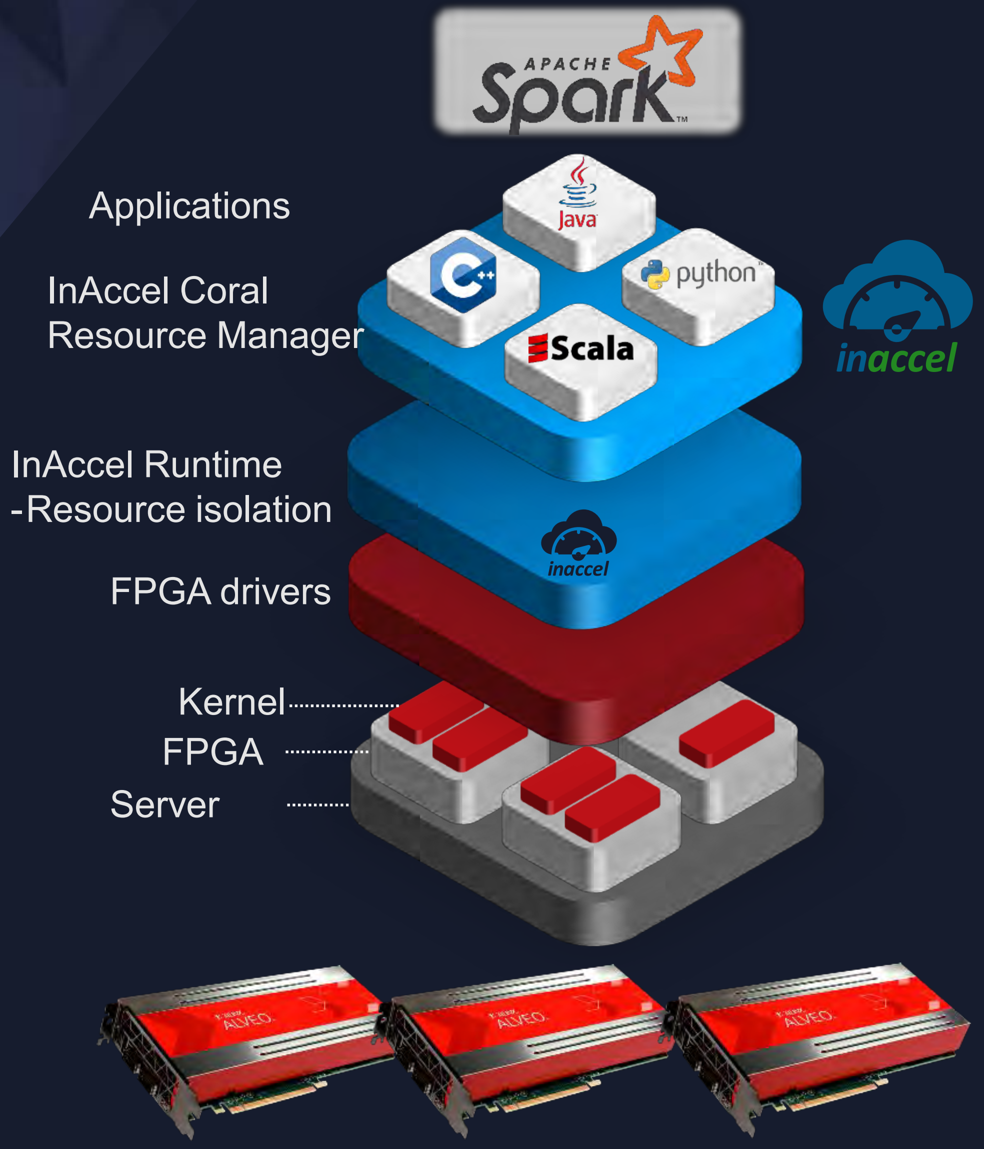
91% PERFORMANCE

69% EASE OF DEPLOYMENT

76% EASE OF PROGRAMMING

Source: Databricks, Apache Spark Survey 2016, Report

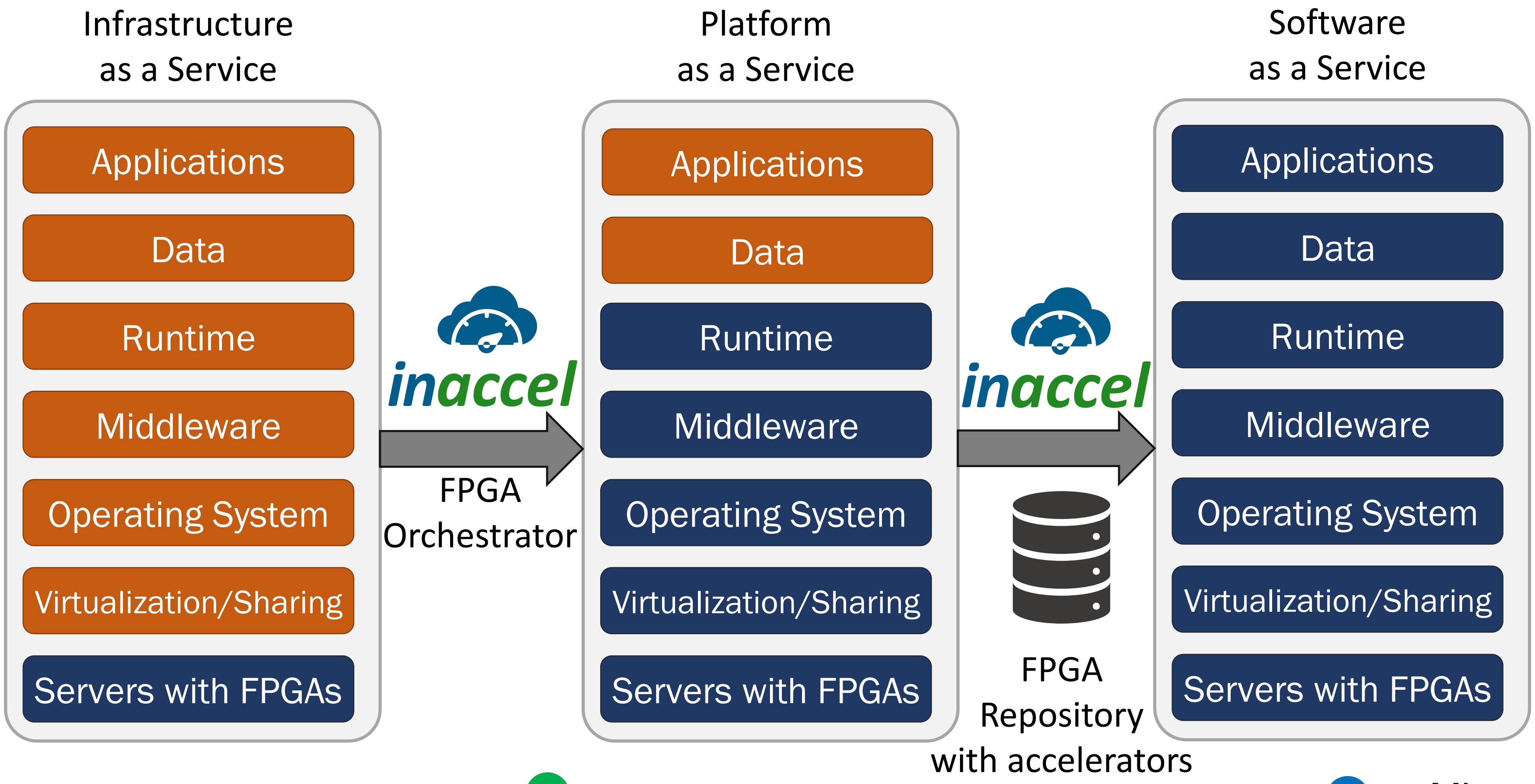
Scalable Orchestrator for FPGA clusters



Automated Deployment, Scaling and Management of FPGA clusters

- Seamless invoking from C/C++, Python, Java and Scala. No need for OpenCL
- Automatic configuration and management of the FPGA **bitstreams** and memory
- Seamless **resource management** of the FPGA cluster from multiple threads/processes/applications/users
- Fully **scalable**: Scale-up (multiple FPGAs per node) and Scale-out (multiple FPGA-based servers over Spark)

PaaS and SaaS for FPGA clusters



Infrastructure as a Service

- Applications
- Data
- Runtime
- Middleware
- Operating System
- Virtualization/Sharing
- Servers with FPGAs

inaccel

FPGA Orchestrator

Platform as a Service

- Applications
- Data
- Runtime
- Middleware
- Operating System
- Virtualization/Sharing
- Servers with FPGAs

inaccel

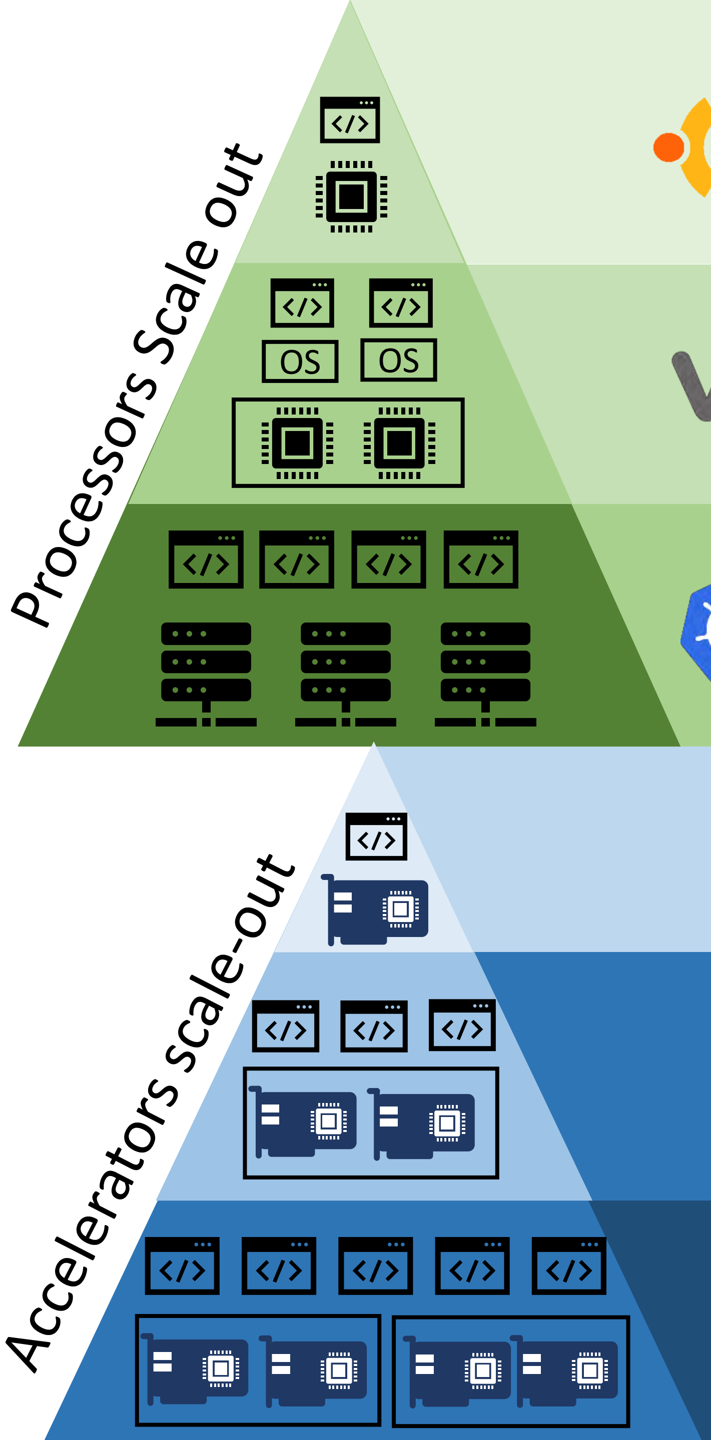
FPGA Repository with accelerators

Software as a Service

- Applications
- Data
- Runtime
- Middleware
- Operating System
- Virtualization/Sharing
- Servers with FPGAs

SW developers
HW Developers

ML engineers
SW developers



Processors Scale out

Linux

vmware®

KVM

kubernetes

docker SWARM

Apache MESOS™

openstack™

Accelerators scale-out

OpenCL

oneAPI

inaccel

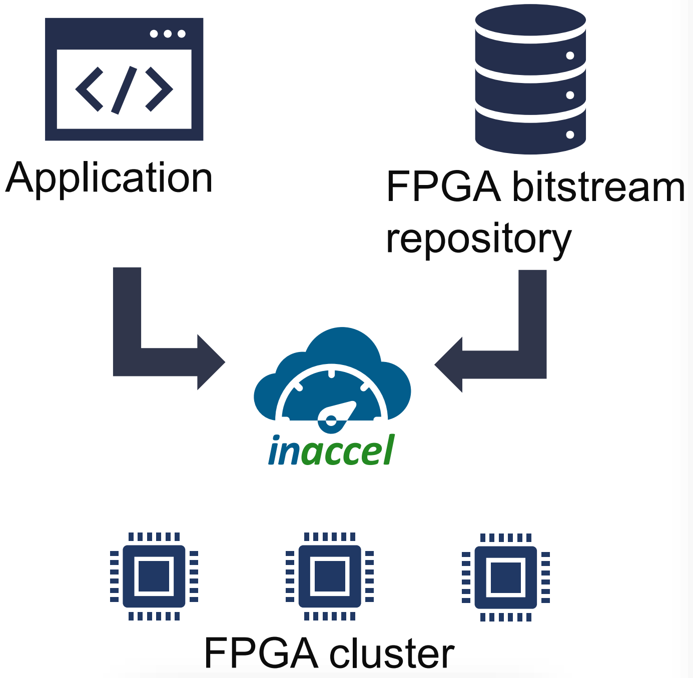
inaccel with kubernetes

www.inaccel.com™

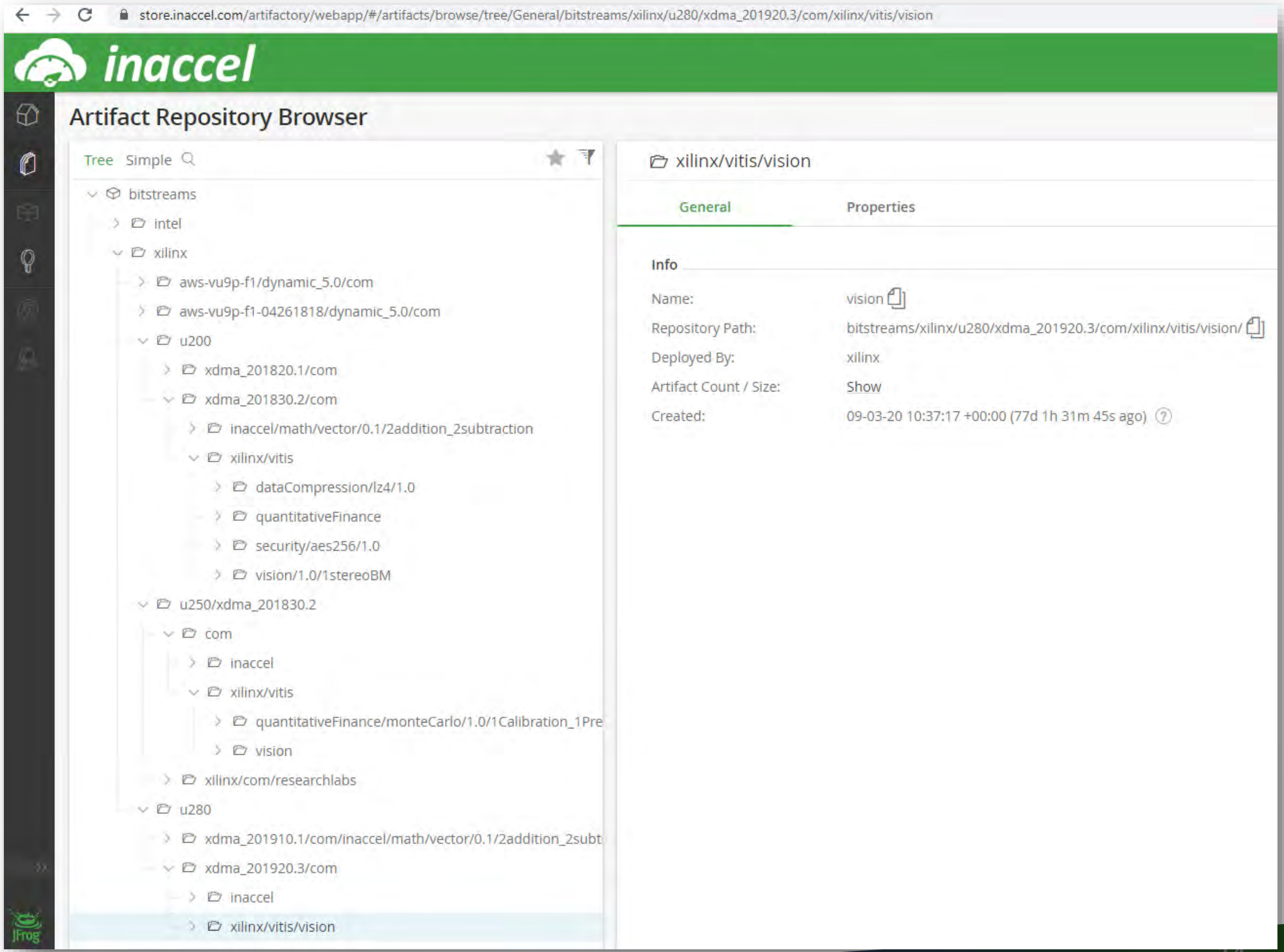
Bitstream repository

- FPGA Resource Manager is integrated with a bitstream repository that is used to store FPGA bitstreams

https://store.inaccel.com



```
inaccel bitstream install [command options]
```



store.inaccel.com/artifactory/webapp/#/artifacts/browse/tree/General/bitstreams/xilinx/u280/xdma_201920.3/com/xilinx/vitis/vision

Artifact Repository Browser

- bitstreams
  - intel
  - xilinx
    - aws-vu9p-f1/dynamic_5.0/com
    - aws-vu9p-f1-04261818/dynamic_5.0/com
    - u200
      - xdma_201820.1/com
      - xdma_201830.2/com
        - inaccel/math/vector/0.1/2addition_2subtraction
        - xilinx/vitis
          - dataCompression/lz4/1.0
          - quantitativeFinance
          - security/aes256/1.0
          - vision/1.0/1stereoBM
    - u250/xdma_201830.2
      - com
        - inaccel
        - xilinx/vitis
          - quantitativeFinance/monteCarlo/1.0/1Calibration_1Pre
          - vision
      - xilinx/com/researchlabs
    - u280
      - xdma_201910.1/com/inaccel/math/vector/0.1/2addition_2subt
      - xdma_201920.3/com
        - inaccel
        - xilinx/vitis/vision

xilinx/vitis/vision

Info

Name: vision

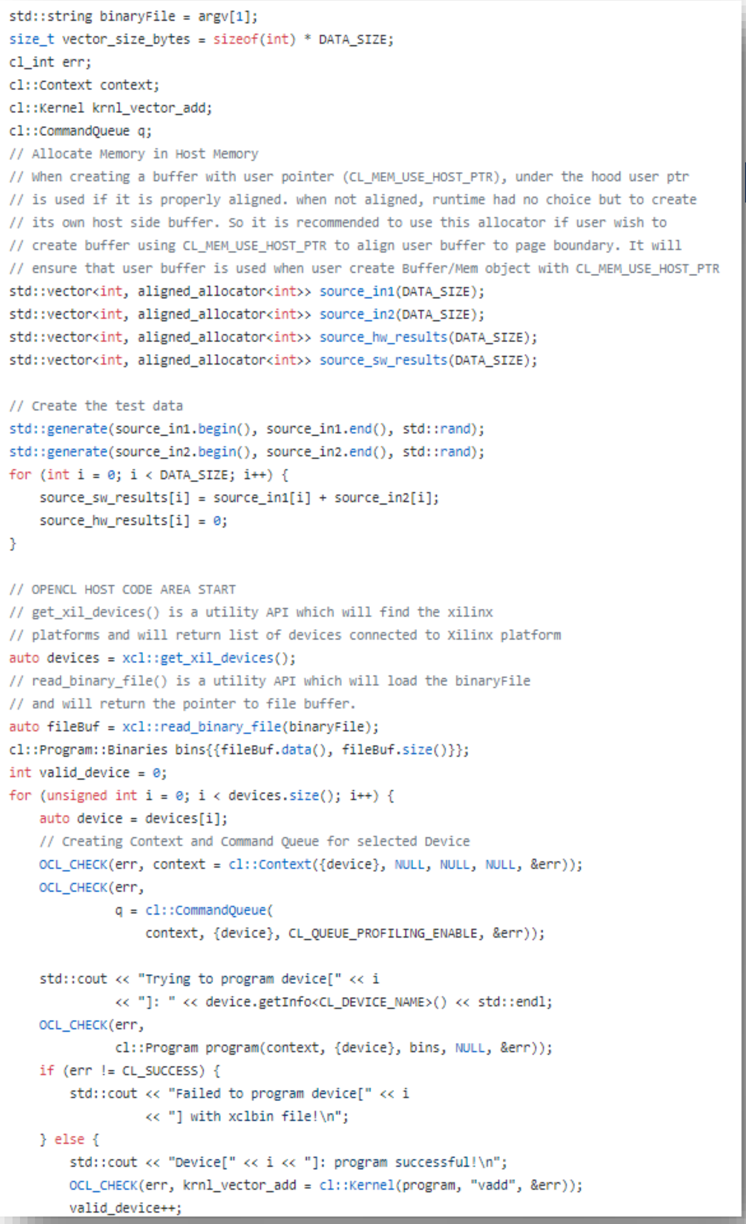
Repository Path: bitstreams/xilinx/u280/xdma_201920.3/com/xilinx/vitis/vision/

Deployed By: xilinx

Artifact Count / Size: Show

Created: 09-03-20 10:37:17 +00:00 (77d 1h 31m 45s ago)

Simple invoking, deployment

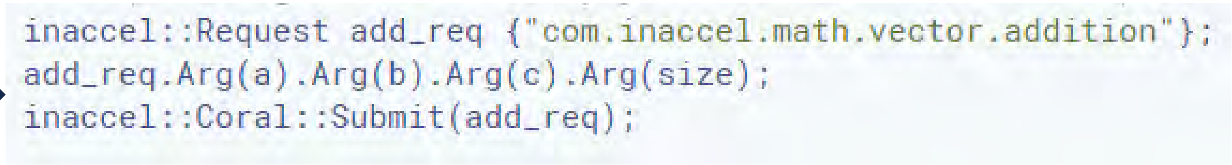


```
std::string binaryFile = argv[1];
size_t vector_size_bytes = sizeof(int) * DATA_SIZE;
cl_int err;
cl::Context context;
cl::Kernel krnl_vector_add;
cl::CommandQueue q;
// Allocate Memory in Host Memory
// When creating a buffer with user pointer (CL_MEM_USE_HOST_PTR), under the hood user ptr
// is used if it is properly aligned. when not aligned, runtime had no choice but to create
// its own host side buffer. So it is recommended to use this allocator if user wish to
// create buffer using CL_MEM_USE_HOST_PTR to align user buffer to page boundary. It will
// ensure that user buffer is used when user create Buffer/Mem object with CL_MEM_USE_HOST_PTR
std::vector<int, aligned_allocator<int>> source_in1(DATA_SIZE);
std::vector<int, aligned_allocator<int>> source_in2(DATA_SIZE);
std::vector<int, aligned_allocator<int>> source_hw_results(DATA_SIZE);
std::vector<int, aligned_allocator<int>> source_sw_results(DATA_SIZE);

// Create the test data
std::generate(source_in1.begin(), source_in1.end(), std::rand);
std::generate(source_in2.begin(), source_in2.end(), std::rand);
for (int i = 0; i < DATA_SIZE; i++) {
    source_sw_results[i] = source_in1[i] + source_in2[i];
    source_hw_results[i] = 0;
}

// OPENCL HOST CODE AREA START
// get_xil_devices() is a utility API which will find the xilinx
// platforms and will return list of devices connected to Xilinx platform
auto devices = xcl::get_xil_devices();
// read_binary_file() is a utility API which will load the binaryFile
// and will return the pointer to file buffer.
auto fileBuf = xcl::read_binary_file(binaryFile);
cl::Program::Binaries bins{{fileBuf.data(), fileBuf.size()}};
int valid_device = 0;
for (unsigned int i = 0; i < devices.size(); i++) {
    auto device = devices[i];
    // Creating Context and Command Queue for selected Device
    OCL_CHECK(err, context = cl::Context({device}, NULL, NULL, NULL, &err));
    OCL_CHECK(err,
              q = cl::CommandQueue(
                  context, {device}, CL_QUEUE_PROFILING_ENABLE, &err));

    std::cout << "Trying to program device[" << i
              << "]: " << device.getInfo<CL_DEVICE_NAME>() << std::endl;
    OCL_CHECK(err,
              cl::Program program(context, {device}, bins, NULL, &err));
    if (err != CL_SUCCESS) {
        std::cout << "Failed to program device[" << i
                  << "] with xclbin file!\n";
    } else {
        std::cout << "Device[" << i << "]: program successful!\n";
        OCL_CHECK(err, krnl_vector_add = cl::Kernel(program, "vadd", &err));
        valid_device++;
```

No need for OpenCL

No need to allocate buffers
No need to specify bitstreams
No need to program specific device

inaccel



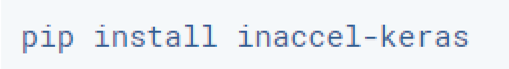
```
inaccel::Request add_req {"com.inaccel.math.vector.addition"};
add_req.Arg(a).Arg(b).Arg(c).Arg(size);
inaccel::Coral::Submit(add_req);
```

- Much simpler invoking
- Software-alike function invoking
- No need for OpenCL directives
- Same API for C/C++, Java, Python
- Native API

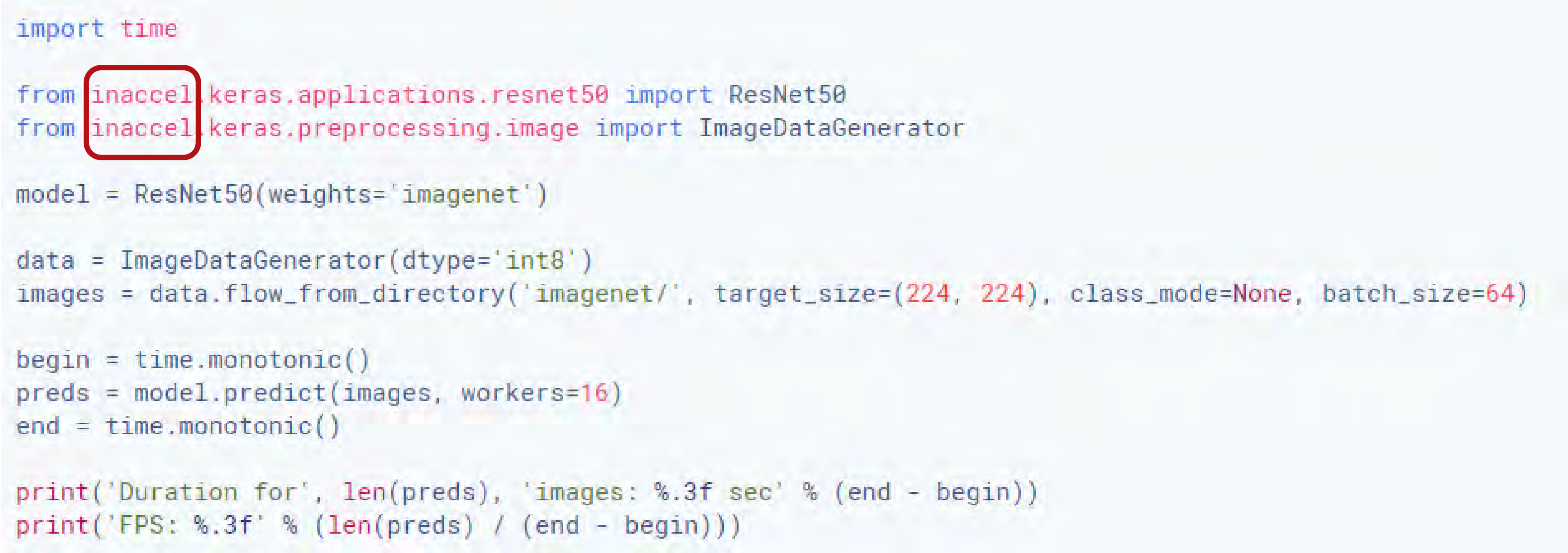
https://github.com/Xilinx/Vitis_Accel_Examples/blob/master/hello_world/src/host.cpp

Keras Deployment on Alveo cards

> **Easy deployment of Keras applications**



```
pip install inaccel-keras
```

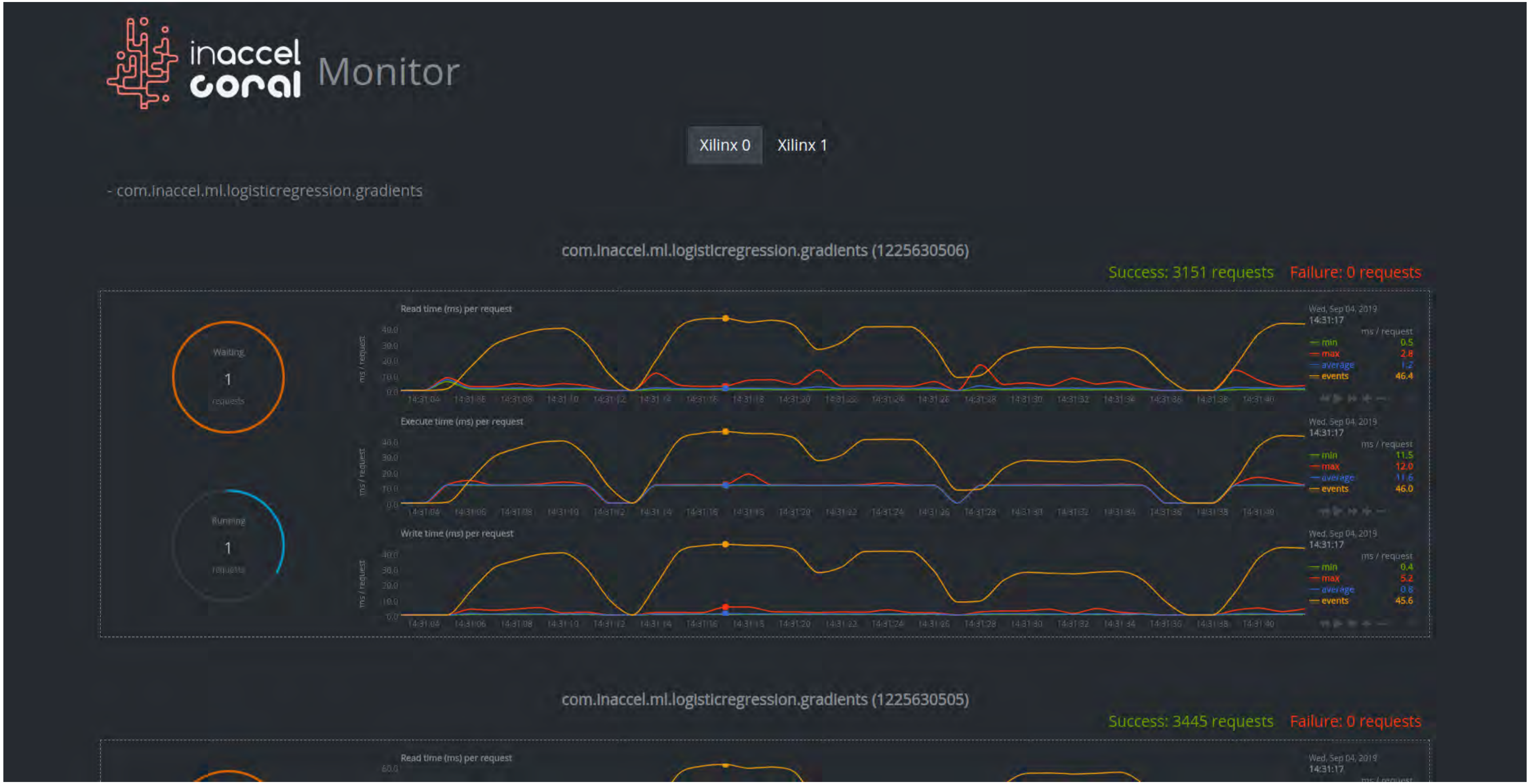


```
import time

from inaccel.keras.applications.resnet50 import ResNet50
from inaccel.keras.preprocessing.image import ImageDataGenerator

model = ResNet50(weights='imagenet')

data = ImageDataGenerator(dtype='int8')
images = data.flow_from_directory('imagenet/', target_size=(224, 224), class_mode=None, batch_size=64)

begin = time.monotonic()
preds = model.predict(images, workers=16)
end = time.monotonic()

print('Duration for', len(preds), 'images: %.3f sec' % (end - begin))
print('FPS: %.3f' % (len(preds) / (end - begin)))
```

2897 fps on U250

https://docs.inaccel.com/project/keras/

Graphical monitoring tool



inaccel coral Monitor

Xilinx 0 Xilinx 1

- com.inaccel.ml.logisticregression.gradients

com.inaccel.ml.logisticregression.gradients (1225630506)

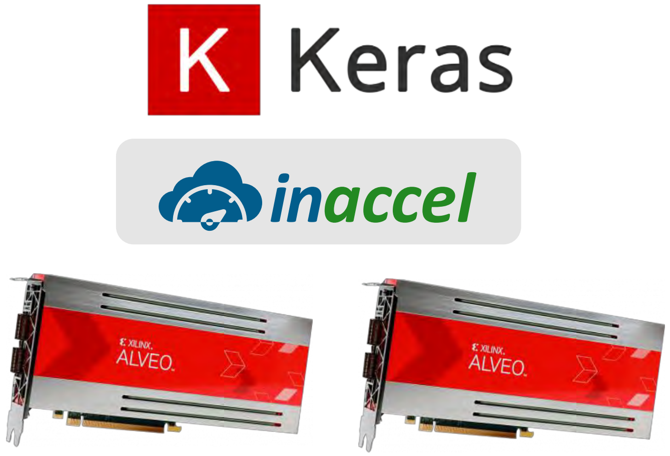
Success: 3151 requests Failure: 0 requests

com.inaccel.ml.logisticregression.gradients (1225630505)

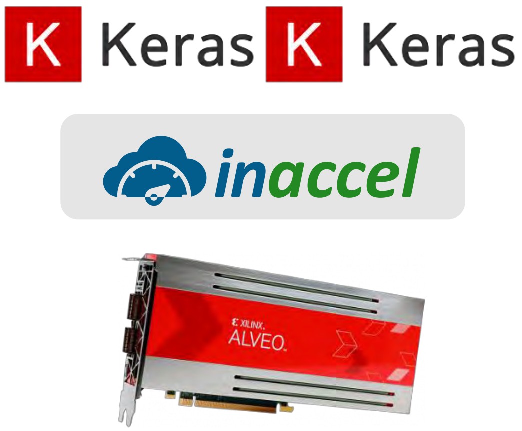
Success: 3445 requests Failure: 0 requests

Quantized ResNet50 on multiple Alveo cards

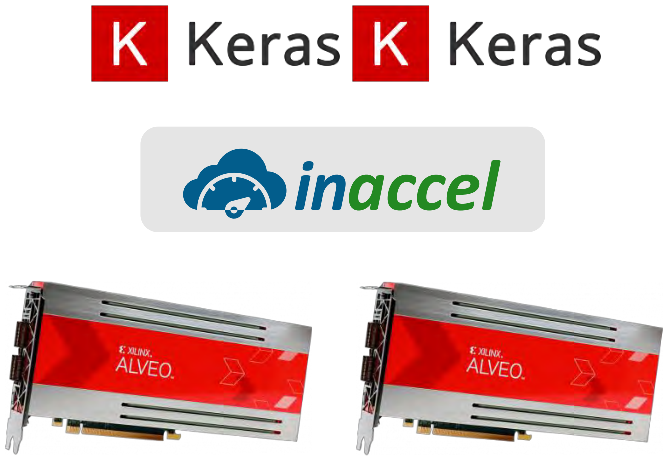
1 Application => 2 Alveo



2 Applications => 1 Alveo



2 Applications => 2 Alveo



https://github.com/Xilinx/ResNet50-PYNQ

Scaling Keras to 2 Alveo cards

> **Same applications => Instant scaling**

2870 fps on 1 U250

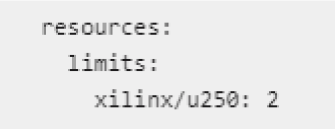
```
[keras/examples]$ 
```

8x fast forward

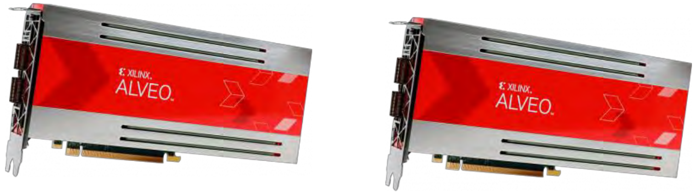
Scaling Keras to 2 Alveo cards

- Same applications => Instant scaling

3681 fps on 2x U250



```
resources:
  limits:
    xilinx/u250: 2
```



```
[keras/examples]$ 
```

8x fast forward

2 Applications in a single Alveo cards

> **Same applications => Instant scaling**

2886 fps on 1x U250

```
[keras/examples]$ 
```

8x fast forward

2 Applications scaling to 2 Alveo cards

> Same applications => Instant scaling

4851 fps on 2x U250

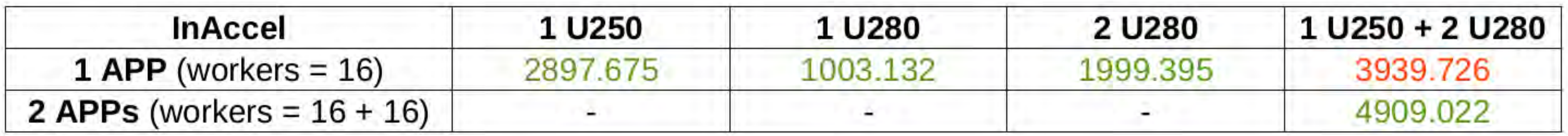
InAccel	1 U250	2 U250
1 APP (workers = 16)	2870.71	3681.413
2 APPs (workers = 16 + 16)	2886.45	4851.603

```
[keras/examples]$ 
```

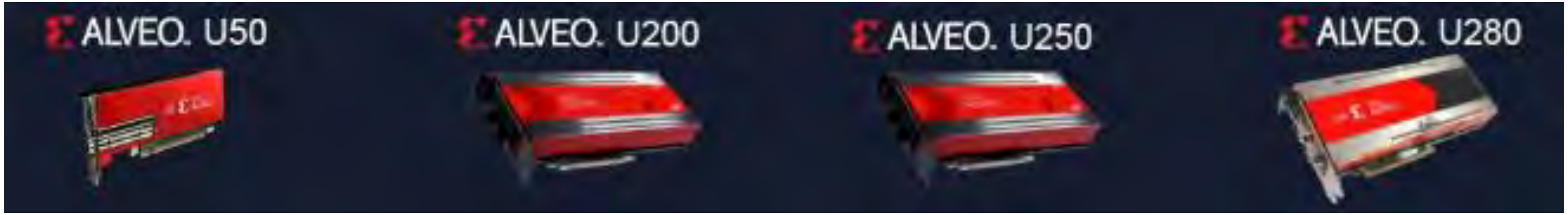
8x fast forward

Heterogeneous deployment

- InAccel FPGA orchestrator is platform agnostic
- You can deploy your applications to heterogeneous Alveo clusters



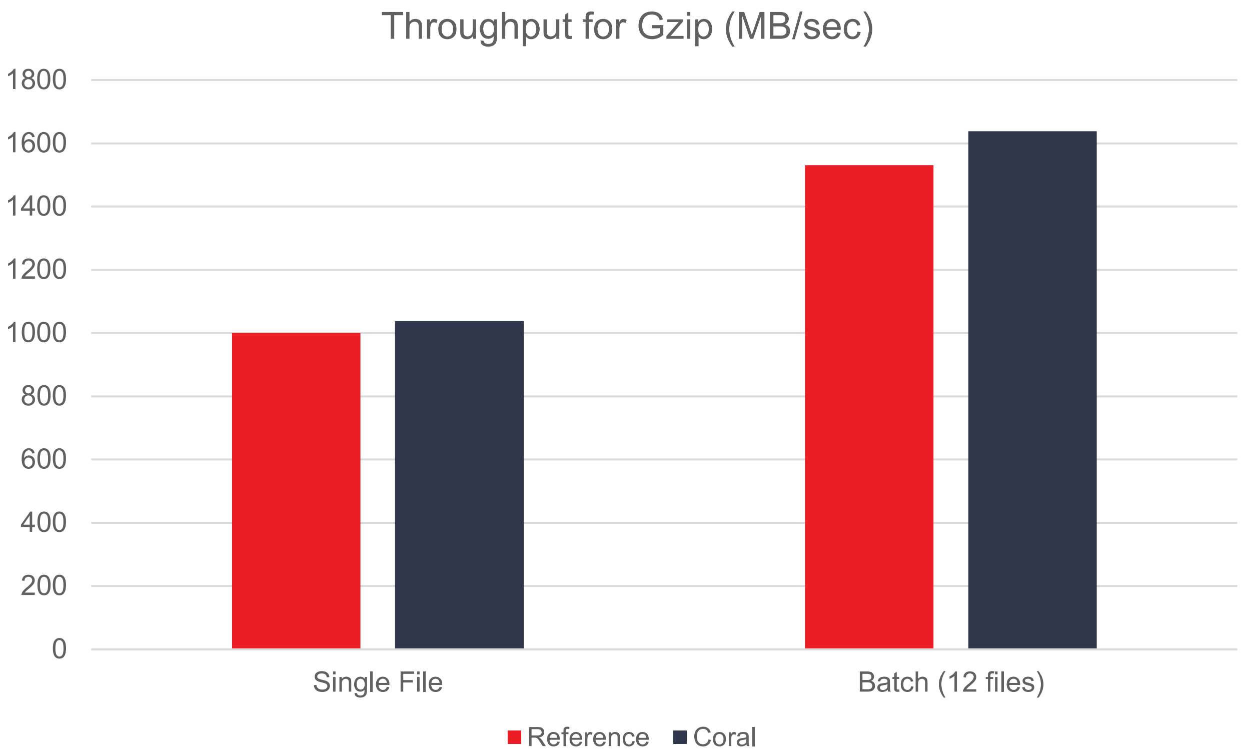
InAccel	1 U250	1 U280	2 U280	1 U250 + 2 U280
1 APP (workers = 16)	2897.675	1003.132	1999.395	3939.726
2 APPs (workers = 16 + 16)	-	-	-	4909.022



Throughput for U280 is indicative and is still in beta version

Zero overhead, Improved Throughput

- Zero overhead
- Improved Performance
- Instant scalability
- Fully virtualization
- Simpler programming

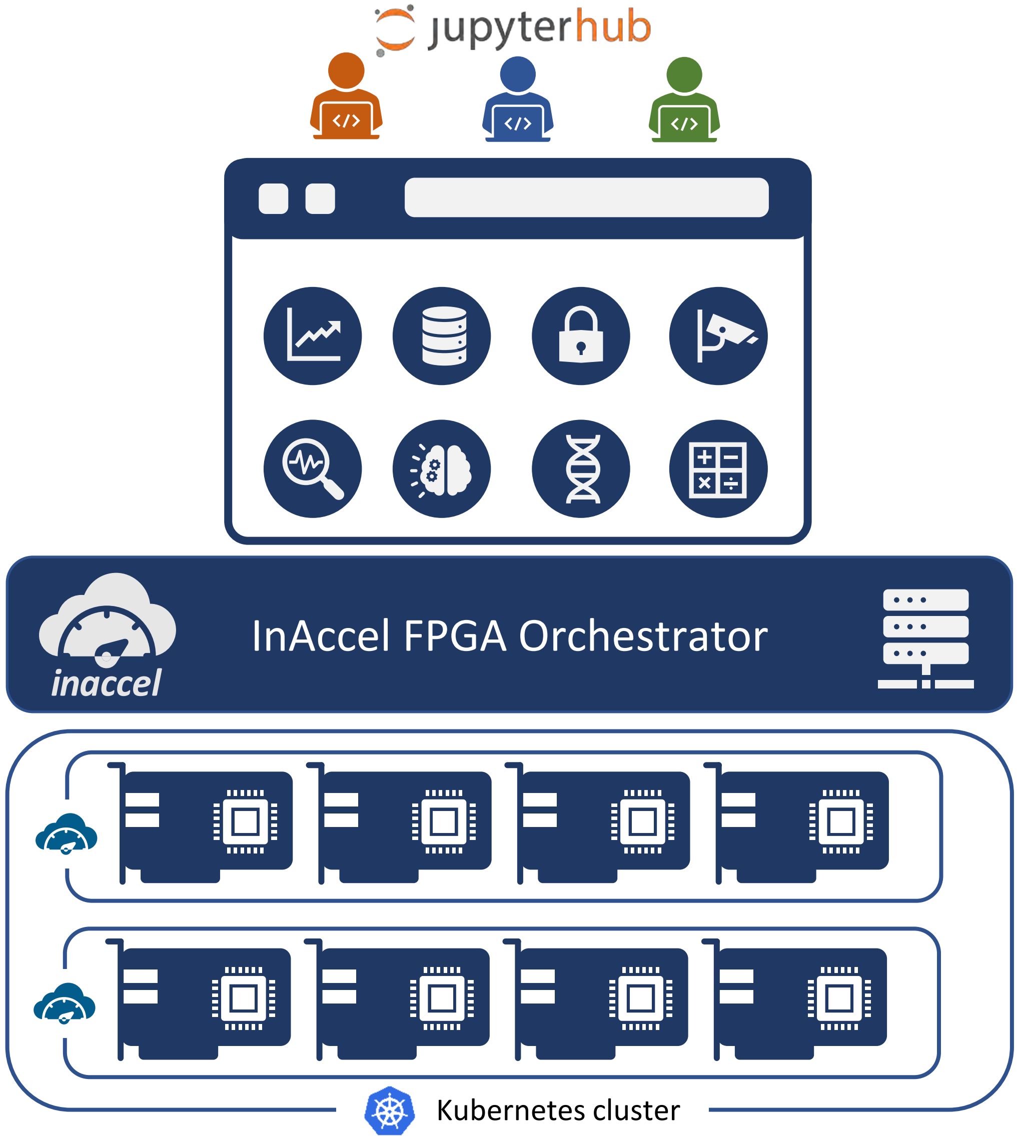


https://github.com/Xilinx/Applications/tree/master/GZip

Multi-tenant Vitis deployment

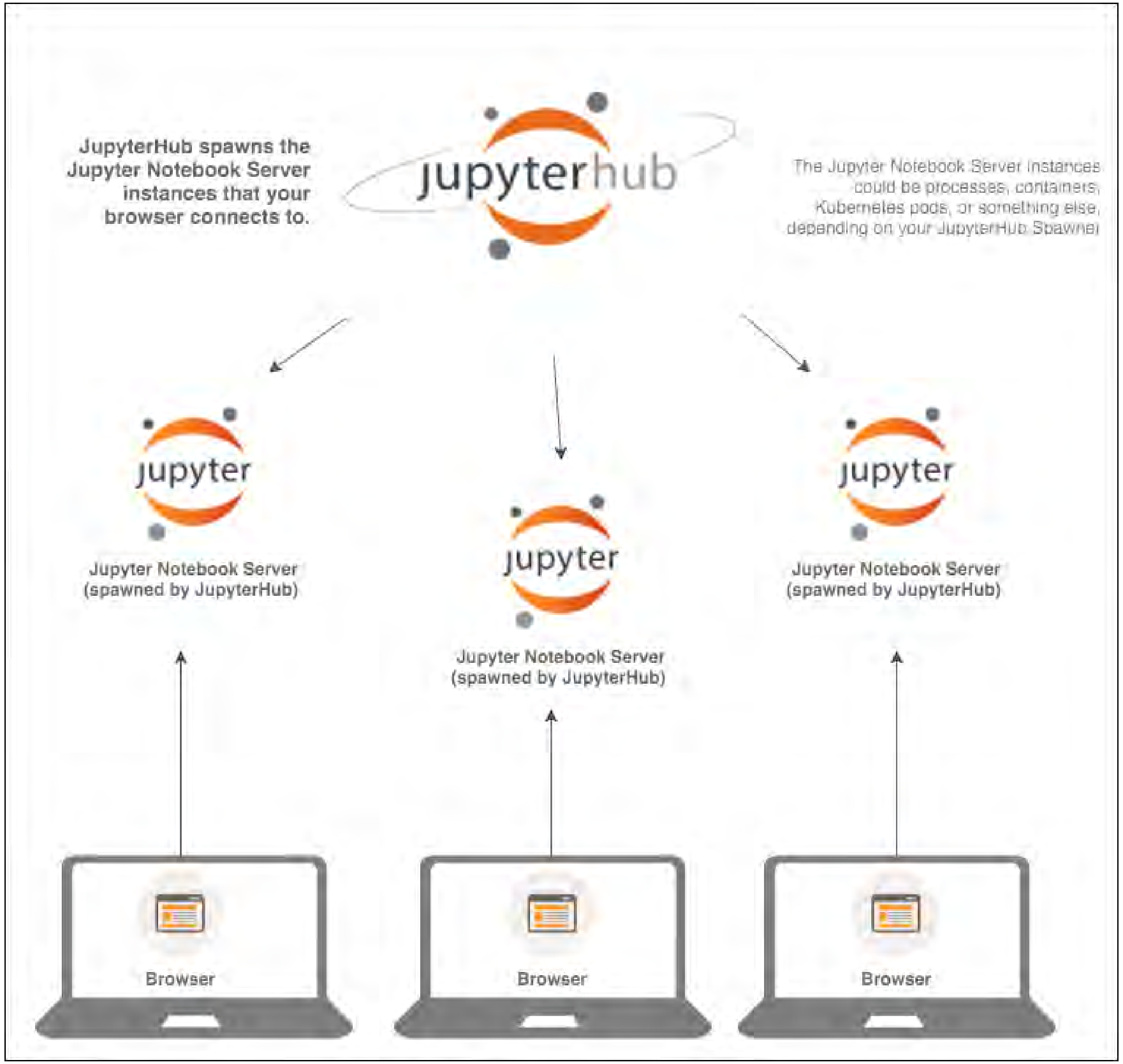
- Run Vitis from browser
- Fully compatible with any Vitis library
- Multi-tenant, multiple applications
- Scalable deployment

https://labs.inaccel.com:8000/



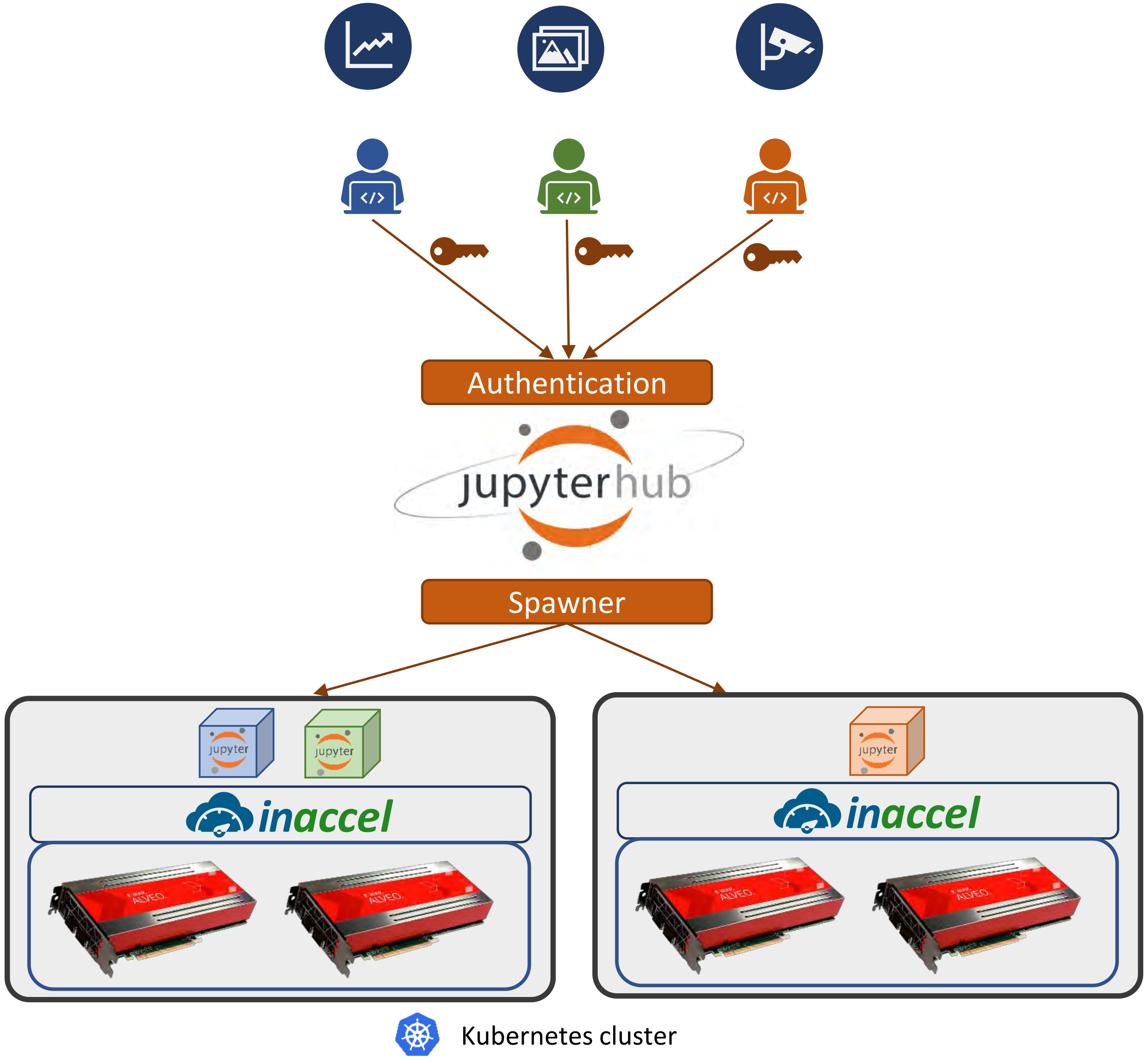
Jupyter - JupyterHub

- > **Deploy and run your FPGA-accelerated applications using Jupyter Notebooks**
- > **InAccel manager allows the instant deployment of FPGAs through HupyterHub**

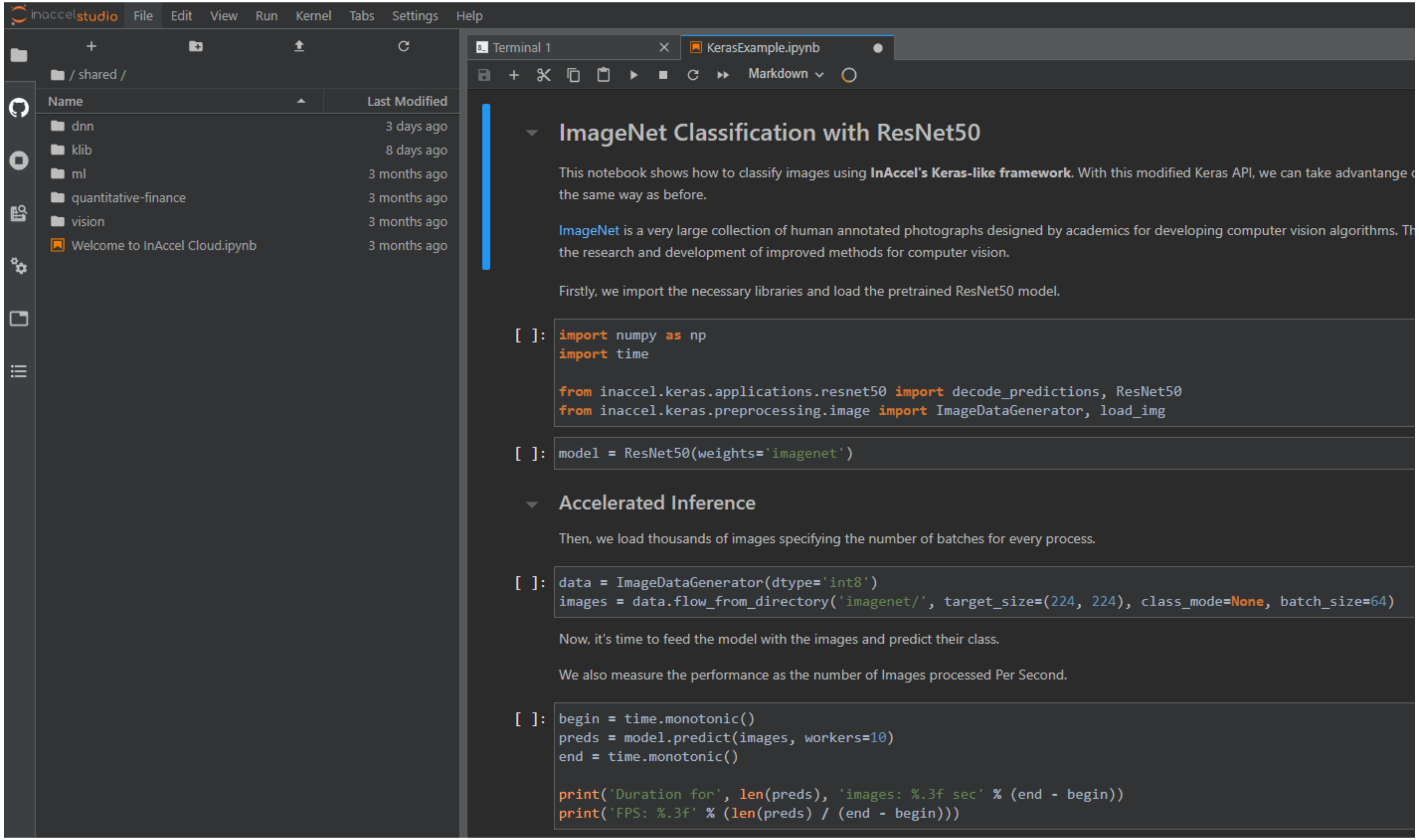

JupyterHub spawns the Jupyter Notebook Server instances that your browser connects to.
jupyterhub
The Jupyter Notebook Server instances could be processes, containers, Kubernetes pods, or something else, depending on your JupyterHub Spawner
jupyter
Jupyter Notebook Server (spawned by JupyterHub)
jupyter
Jupyter Notebook Server (spawned by JupyterHub)
jupyter
Jupyter Notebook Server (spawned by JupyterHub)
Browser
Browser
Browser

JupyterHub on FPGAs

- Instant acceleration of Jupyter Notebooks with zero code-changes
- Offload the most computational intensive tasks on FPGA-based servers



Vitis on Alveo cluster on a browser



Name	Last Modified
dnn	3 days ago
klib	8 days ago
ml	3 months ago
quantitative-finance	3 months ago
vision	3 months ago
Welcome to InAccel Cloud.ipynb	3 months ago

ImageNet Classification with ResNet50

This notebook shows how to classify images using **InAccel's Keras-like framework**. With this modified Keras API, we can take advantange the same way as before.

ImageNet is a very large collection of human annotated photographs designed by academics for developing computer vision algorithms. Th the research and development of improved methods for computer vision.

Firstly, we import the necessary libraries and load the pretrained ResNet50 model.

```
[ ]: import numpy as np
     import time

     from inaccel.keras.applications.resnet50 import decode_predictions, ResNet50
     from inaccel.keras.preprocessing.image import ImageDataGenerator, load_img
```

```
[ ]: model = ResNet50(weights='imagenet')
```

Accelerated Inference

Then, we load thousands of images specifying the number of batches for every process.

```
[ ]: data = ImageDataGenerator(dtype='int8')
     images = data.flow_from_directory('imagenet/', target_size=(224, 224), class_mode=None, batch_size=64)
```

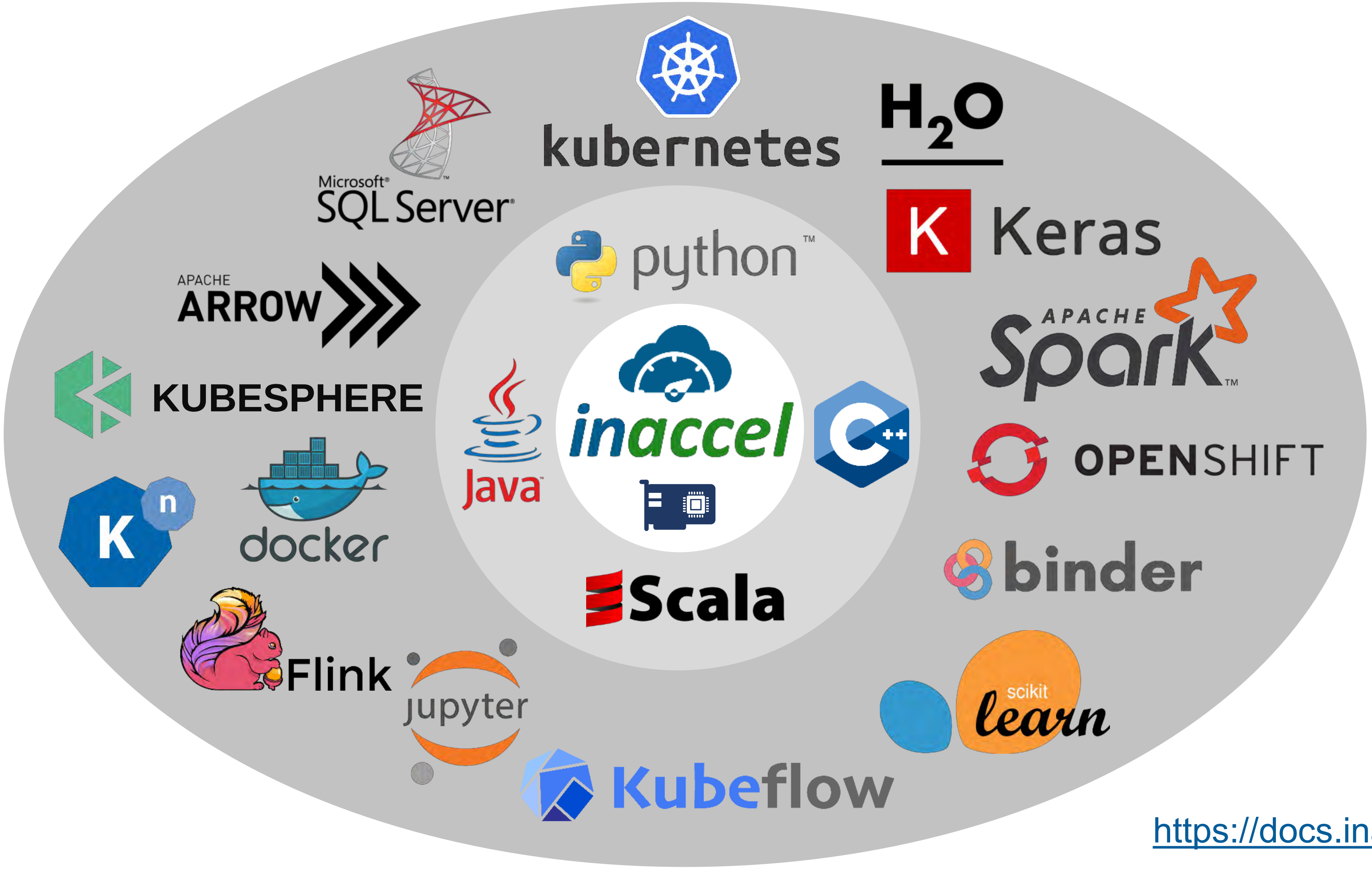
Now, it's time to feed the model with the images and predict their class.

We also measure the performance as the number of Images processed Per Second.

```
[ ]: begin = time.monotonic()
     preds = model.predict(images, workers=10)
     end = time.monotonic()

     print('Duration for', len(preds), 'images: %.3f sec' % (end - begin))
     print('FPS: %.3f' % (len(preds) / (end - begin)))
```

https://youtu.be/FKwvI89dgsg https://labs.inaccel.com:8000/

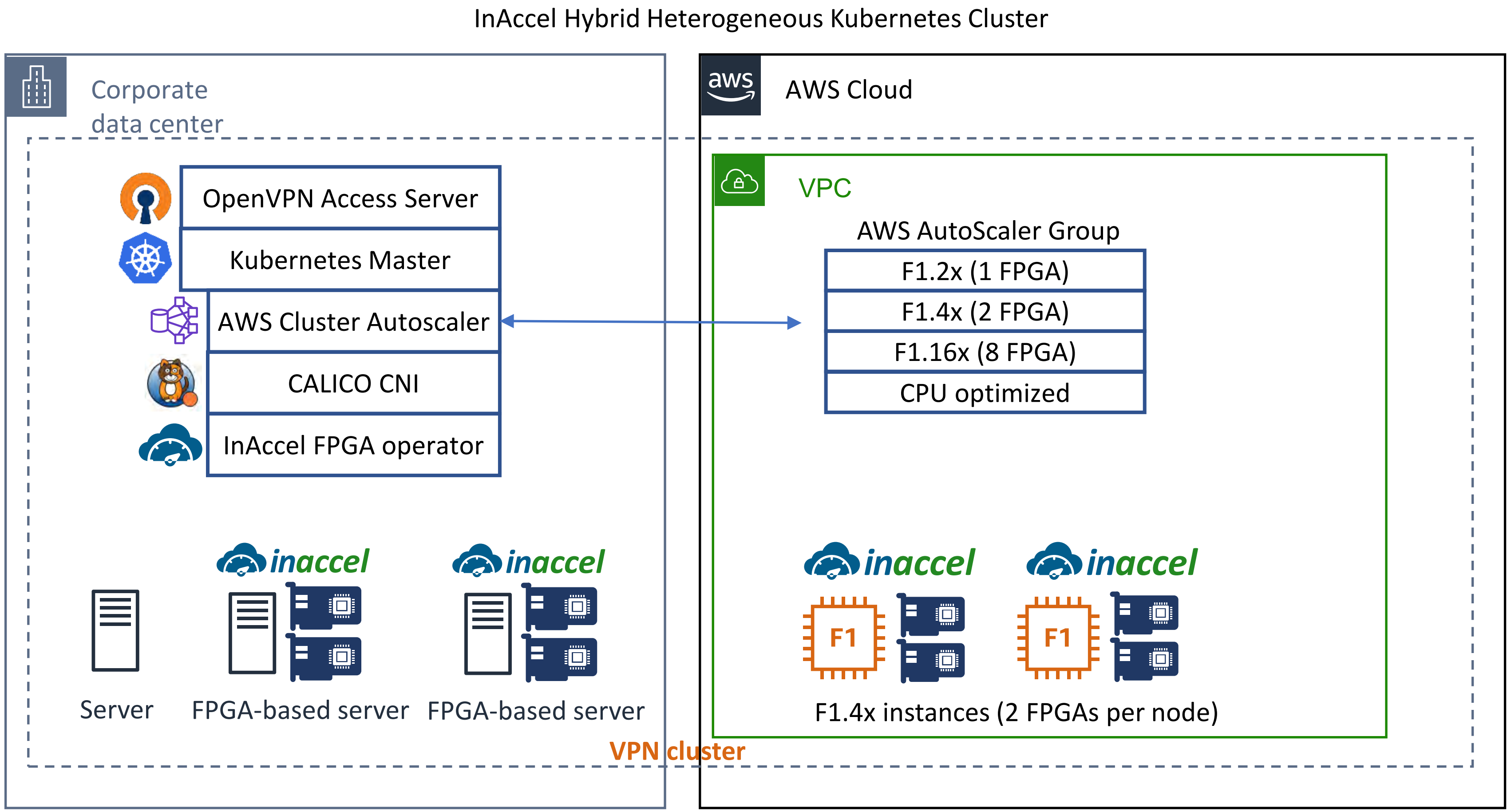
Successful Use cases, Integrations



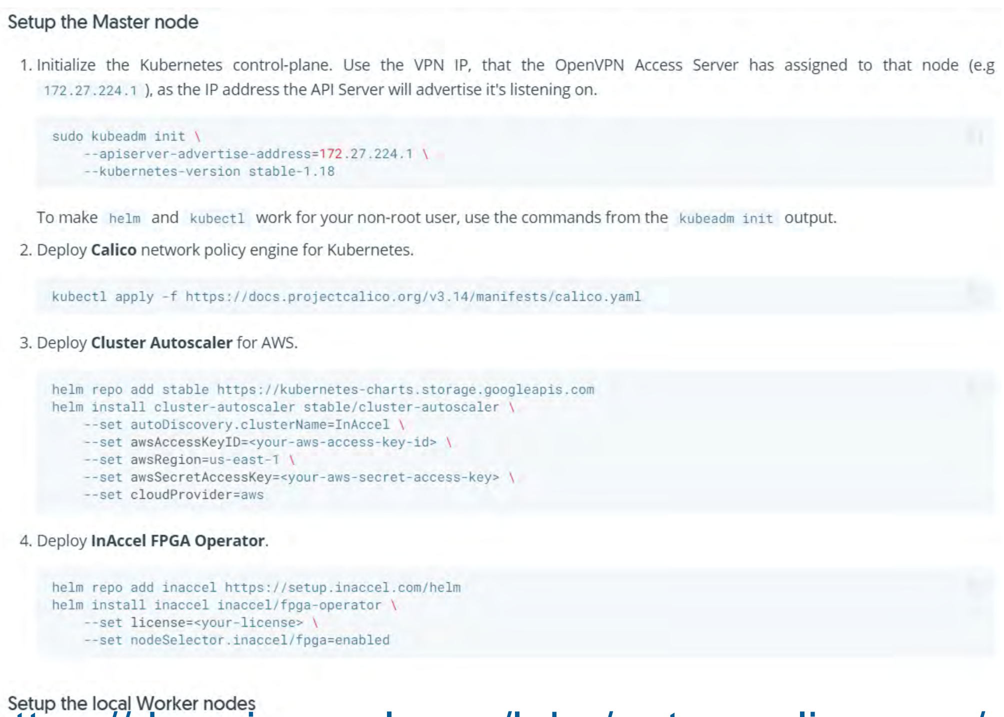
https://docs.inaccel.com/

Auto-scalable deployment

- Starting on prem
- Moving to the cloud
  - Automatically
  - Instantly



Auto-scalable FPGA deployment



Setup the Master node

1. Initialize the Kubernetes control-plane. Use the VPN IP, that the OpenVPN Access Server has assigned to that node (e.g `172.27.224.1`), as the IP address the API Server will advertise it's listening on.

```
sudo kubeadm init \
    --apiserver-advertise-address=172.27.224.1 \
    --kubernetes-version stable-1.18
```

   To make `helm` and `kubectl` work for your non-root user, use the commands from the `kubeadm init` output.

2. Deploy **Calico** network policy engine for Kubernetes.

```
kubectl apply -f https://docs.projectcalico.org/v3.14/manifests/calico.yaml
```

3. Deploy **Cluster Autoscaler** for AWS.

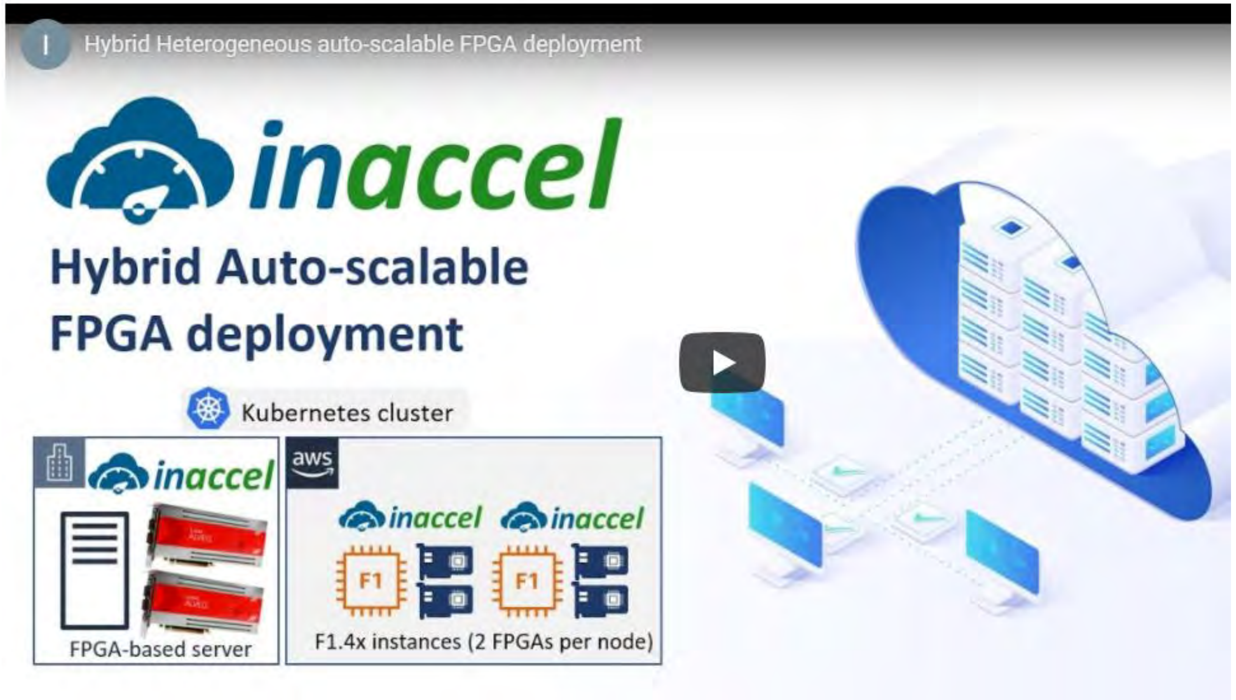
```
helm repo add stable https://kubernetes-charts.storage.googleapis.com
helm install cluster-autoscaler stable/cluster-autoscaler \
    --set autoDiscovery.clusterName=InAccel \
    --set awsAccessKeyID=<your-aws-access-key-id> \
    --set awsRegion=us-east-1 \
    --set awsSecretAccessKey=<your-aws-secret-access-key> \
    --set cloudProvider=aws
```

4. Deploy **InAccel FPGA Operator**.

```
helm repo add inaccel https://setup.inaccel.com/helm
helm install inaccel inaccel/fpga-operator \
    --set license=<your-license> \
    --set nodeSelector.inaccel/fpga=enabled
```

Setup the local Worker nodes

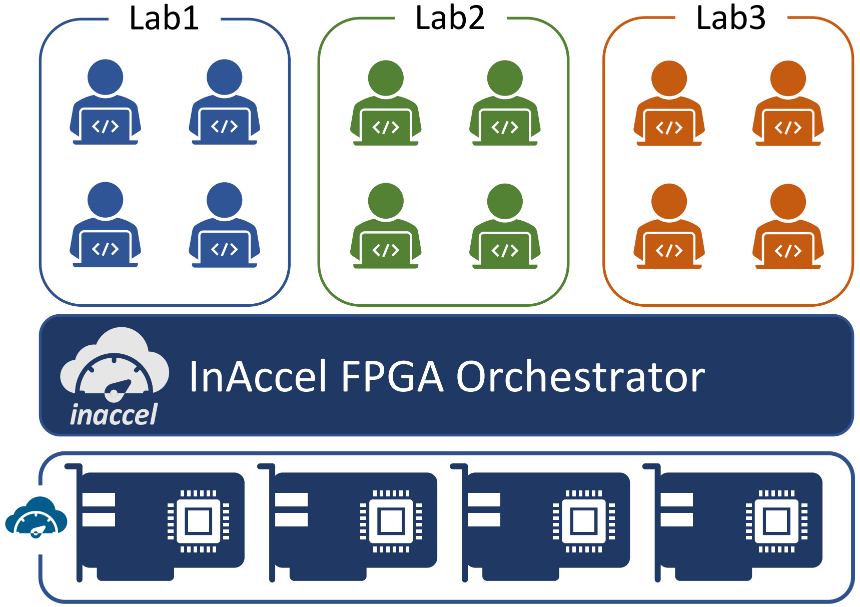
https://docs.inaccel.com/labs/auto-scaling-aws/



https://www.youtube.com/watch?v=CVVyvXY4w5w

Universities

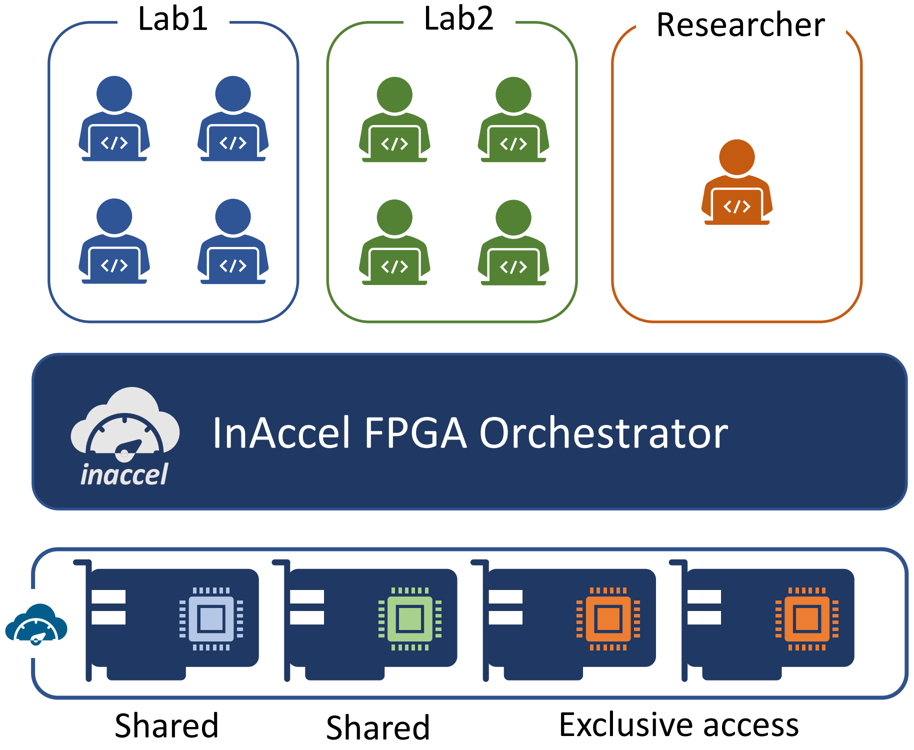
- How do you allow multiple students to share the available FPGAs?
- Many universities have limited number of FPGA cards that want to share with multiple students.
- InAccel FPGA orchestrator allows multiple students to share one or more FPGAs seamlessly.
- It allows students to just invoke the function that want to accelerate and InAccel FPGA manager performs the serialization and the scheduling of the functions to the available FPGA resources.



InAccel FPGA Orchestrator

Universities

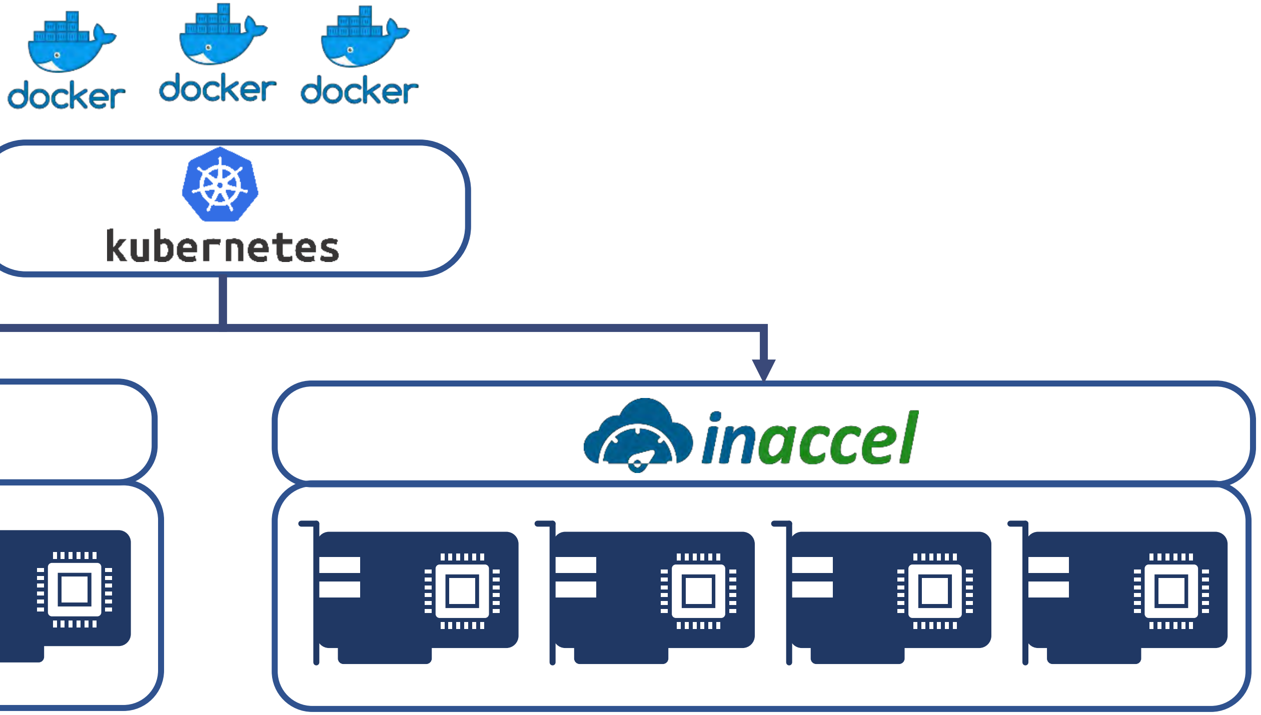
- > **But the researchers want exclusive access**
- > InAccel orchestrator allows to select which FPGA cards will be available for multiple students and which FPGAs can be allocated exclusively to researchers and Ph.D. students (so they can get accurate measurements for their papers).
- > The FPGAs that are shared with multiple students will perform on a best-effort approach (InAccel manager performs the serialization of the requested access) while the researchers have exclusive access to the FPGAs with zero overhead.



InAccel Coral manager - Kubernetes

> **Integrated solution that allows**

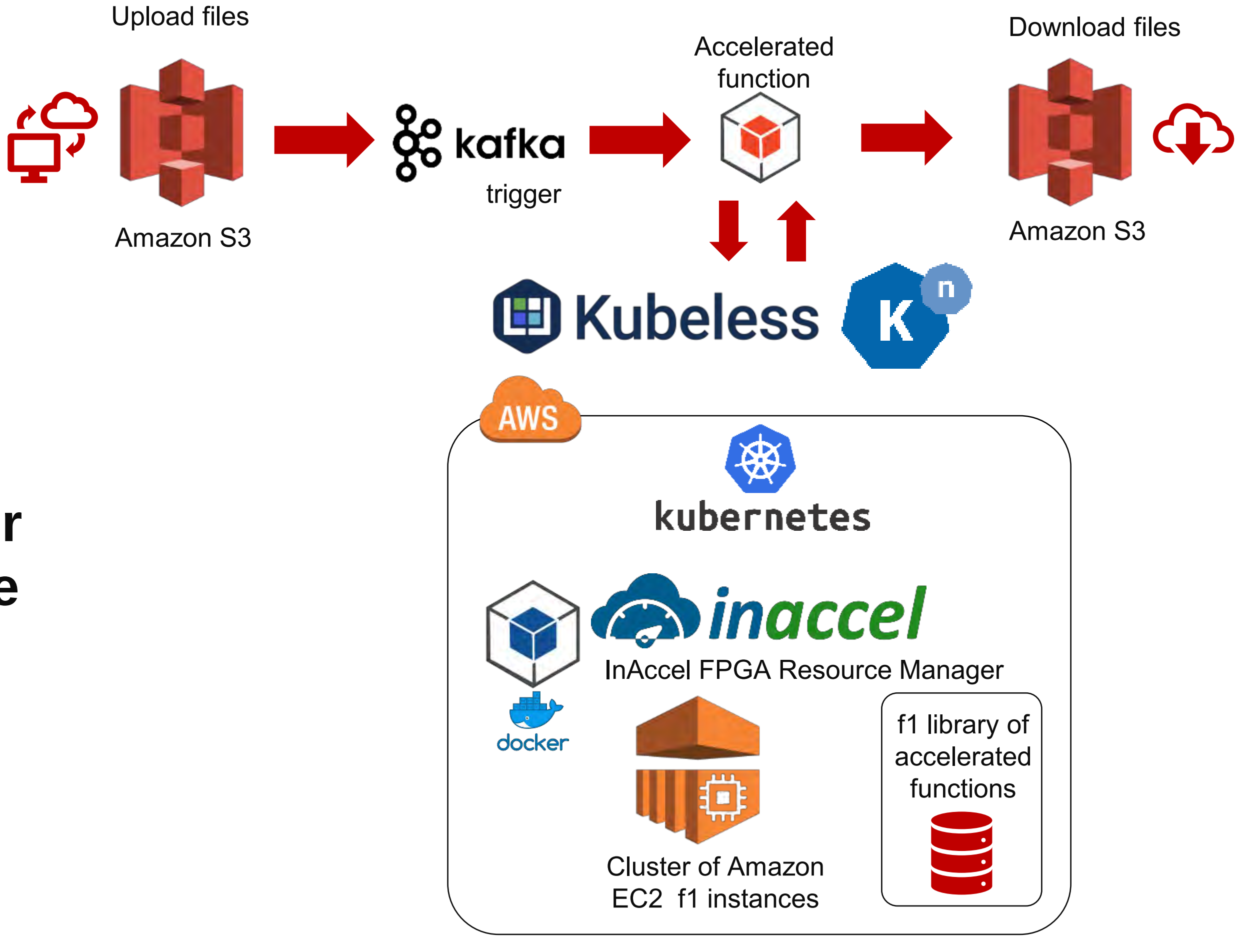
- Scale Up (1, 2, or 8 FPGAs per server)
- Scale Out to multiple servers



Serverless deployment

- **Integrated framework for serverless deployment**
- **Compatible with Kubeless, Knative**

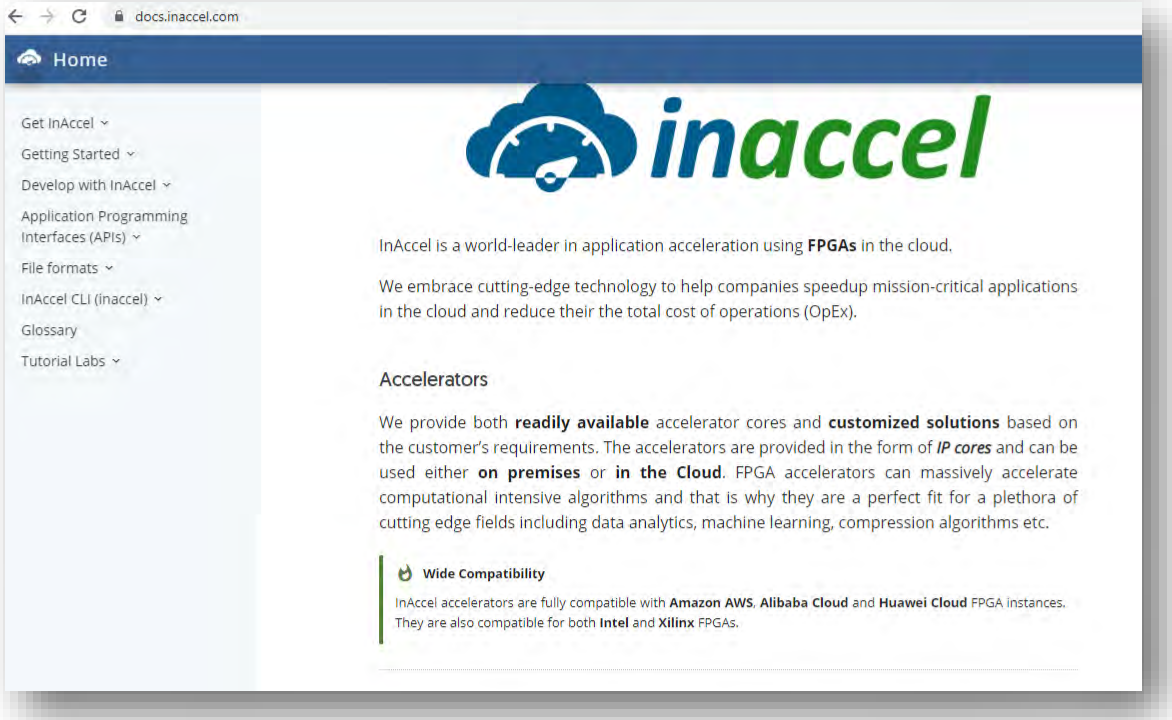
- Users only have to **upload the images** on the S3 bucket and then InAccel's FPGA Manager **automatically deploy the cluster of FPGAs**, process the data and then **store back the results** on the S3 bucket.
- Users do not have to know anything about the FPGA execution.



https://medium.com/@inaccel/fpgas-goes-serverless-on-kubernetes-55c1d39c5e30

Test it on your prem or on your browser

On-prem



InAccel is a world-leader in application acceleration using **FPGAs** in the cloud.

We embrace cutting-edge technology to help companies speedup mission-critical applications in the cloud and reduce their the total cost of operations (OpEx).

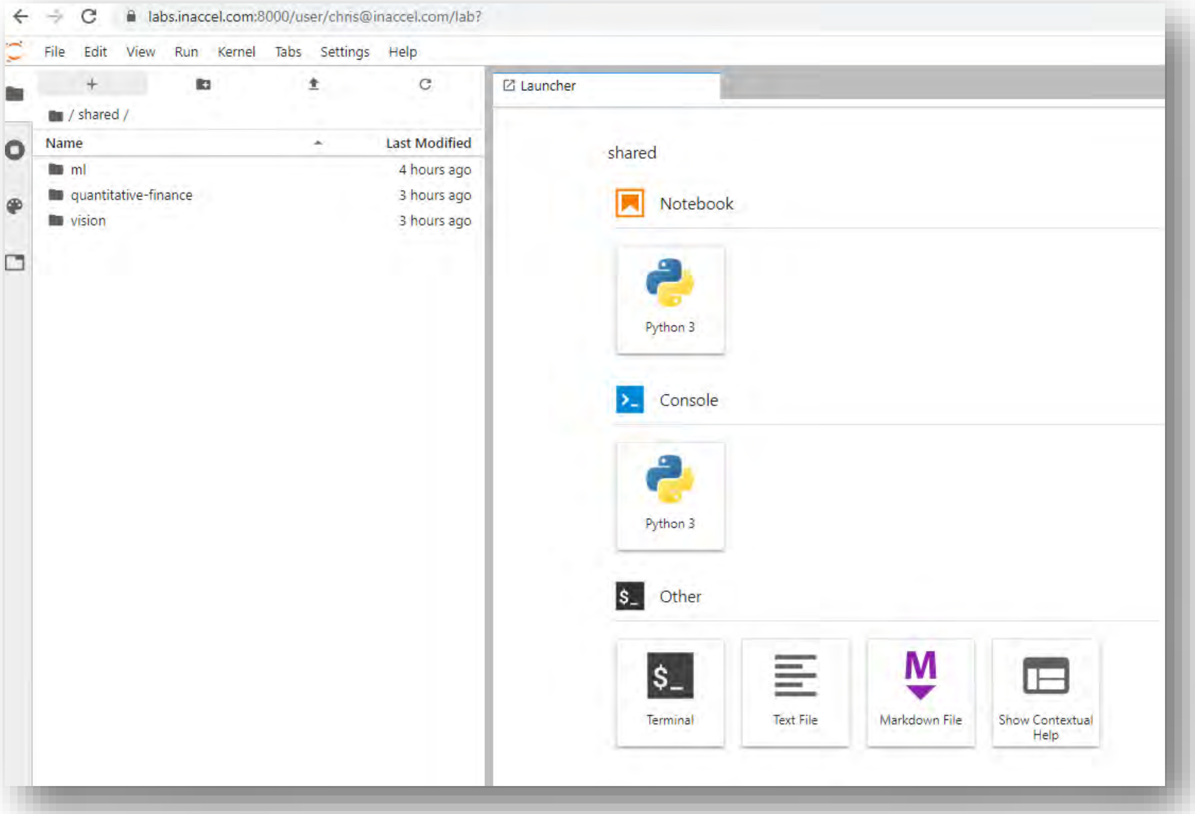
Accelerators

We provide both **readily available** accelerator cores and **customized solutions** based on the customer's requirements. The accelerators are provided in the form of ***IP cores*** and can be used either **on premises** or **in the Cloud**. FPGA accelerators can massively accelerate computational intensive algorithms and that is why they are a perfect fit for a plethora of cutting edge fields including data analytics, machine learning, compression algorithms etc.

> **Wide Compatibility**
>
> InAccel accelerators are fully compatible with **Amazon AWS**, **Alibaba Cloud** and **Huawei Cloud** FPGA instances. They are also compatible for both **Intel** and **Xilinx** FPGAs.

https://docs.inaccel.com/

Online - Browser



Name	Last Modified
ml	4 hours ago
quantitative-finance	3 hours ago
vision	3 hours ago

https://labs.inaccel.com:8000/

InAccel, Inc. Corporate overview

inaccel

- > **Founded in January 2018**
- > **Registered in Delaware, USA**

- > **Membership:**

XILINX ALLIANCE PROGRAM MEMBER

amazon web services Partner Network ADVANCED TECHNOLOGY PARTNER

Alibaba Cloud Partner Network

Registered Technology Partner

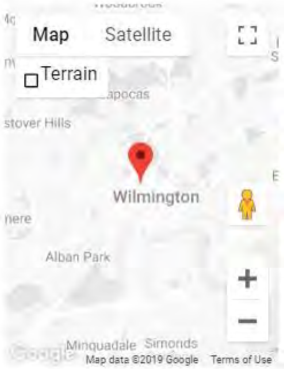
NIMBIX supercomputing made super human™

HUAWEI

Tencent Cloud

Azure

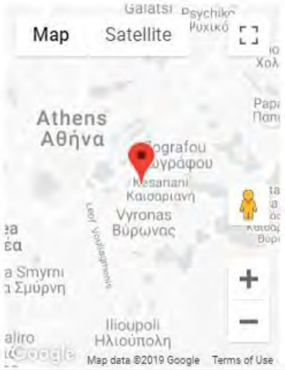
Microsoft for Startups



Headquarters

500 Delaware Ave STE 1, #1960
Wilmington, DE 19801
USA

☎ (+1) 408 260 5724



Design Center

Formionos 47
Kesariani 116 33
Athens, Greece

☎ (+30) 211 1825 436

MORPHEMIC

This project has received funding from the European Union's Horizon 2020 Research and Innovation program under grant agreement No. 871643.

inaccel

Application Acceleration, seamlessly

www.inaccel.com

info@inaccel.com

USA:

500 Delaware Ave STE 1, #1960
Wilmington, DE 19801
USA

Europe (Design Center):

Formionos 47
Kesariani 116 33
Athens, Greece