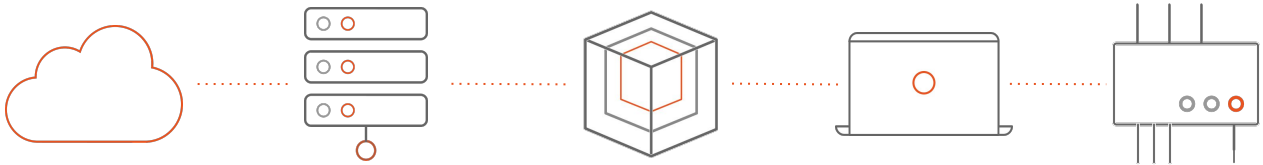
Accelerate your smart-city Edge AI deployments

With open source cloud-native infrastructure

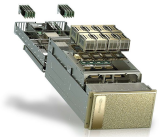
Gustavo Sanchez
Data & AI Solutions Architect
Canonical

Canonical's Ubuntu

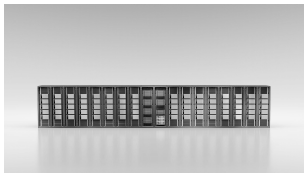
From Cloud, Server, to Desktop and IOT



Bluefield DPUs



NVIDIA H100 /
NVIDIA DGX

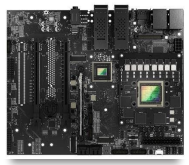


NVIDIA DGX
Superpods

Native integrations
with Ubuntu across
NVIDIA hardware

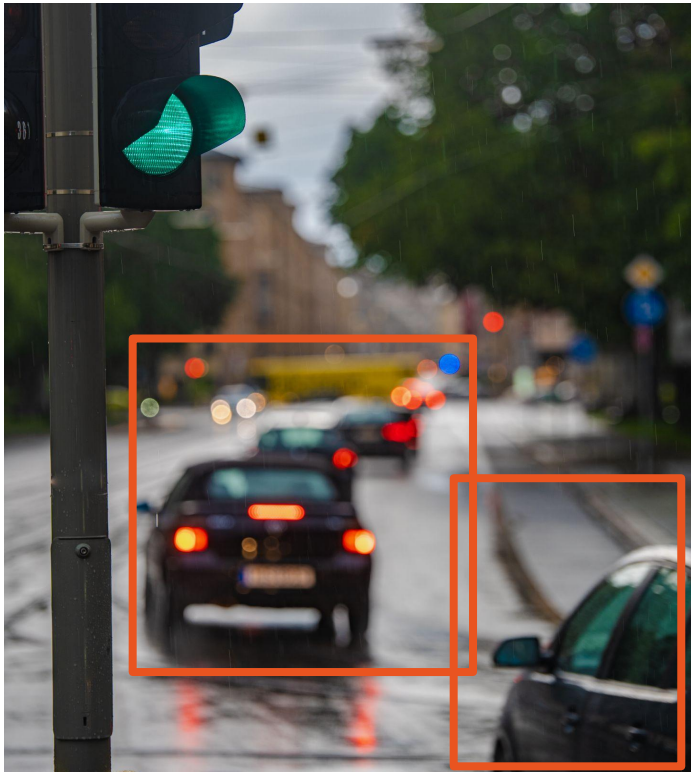
NVIDIA IGX

RTX



NVIDIA Orin

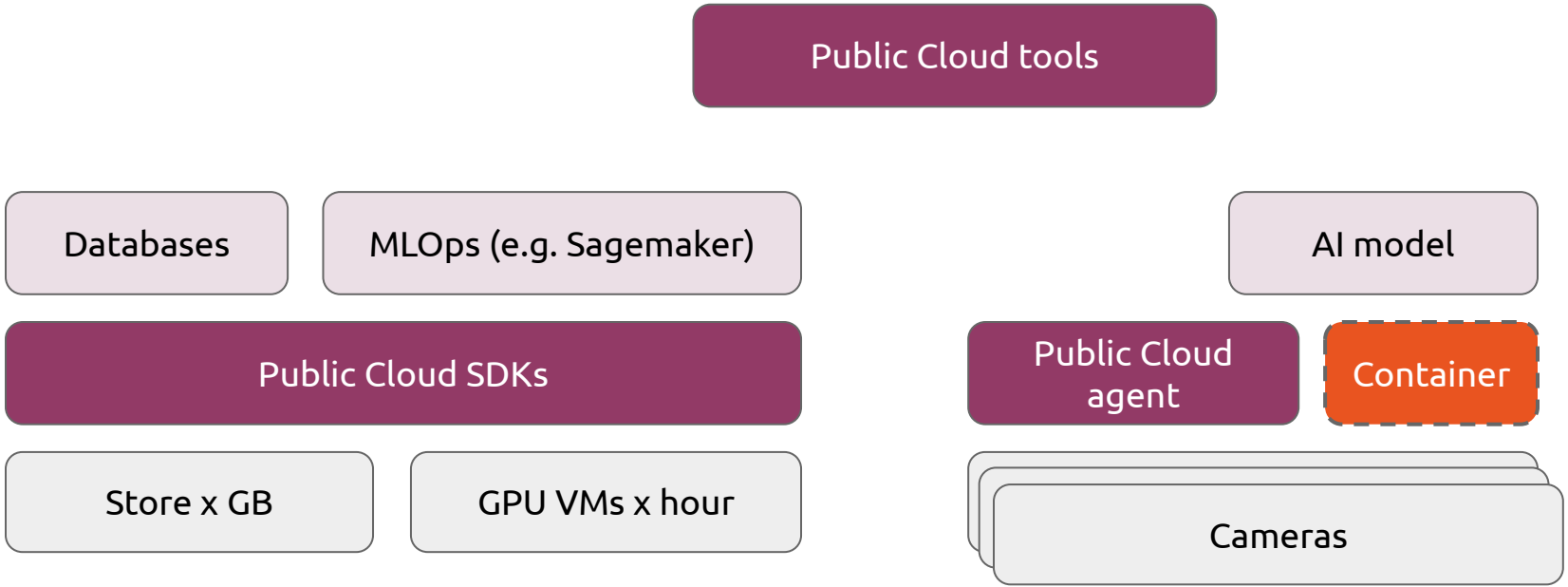
Smart Cameras
AI From Cloud to Edge



AI ML Developer Journey | Challenges

Solution architecture

Baseline



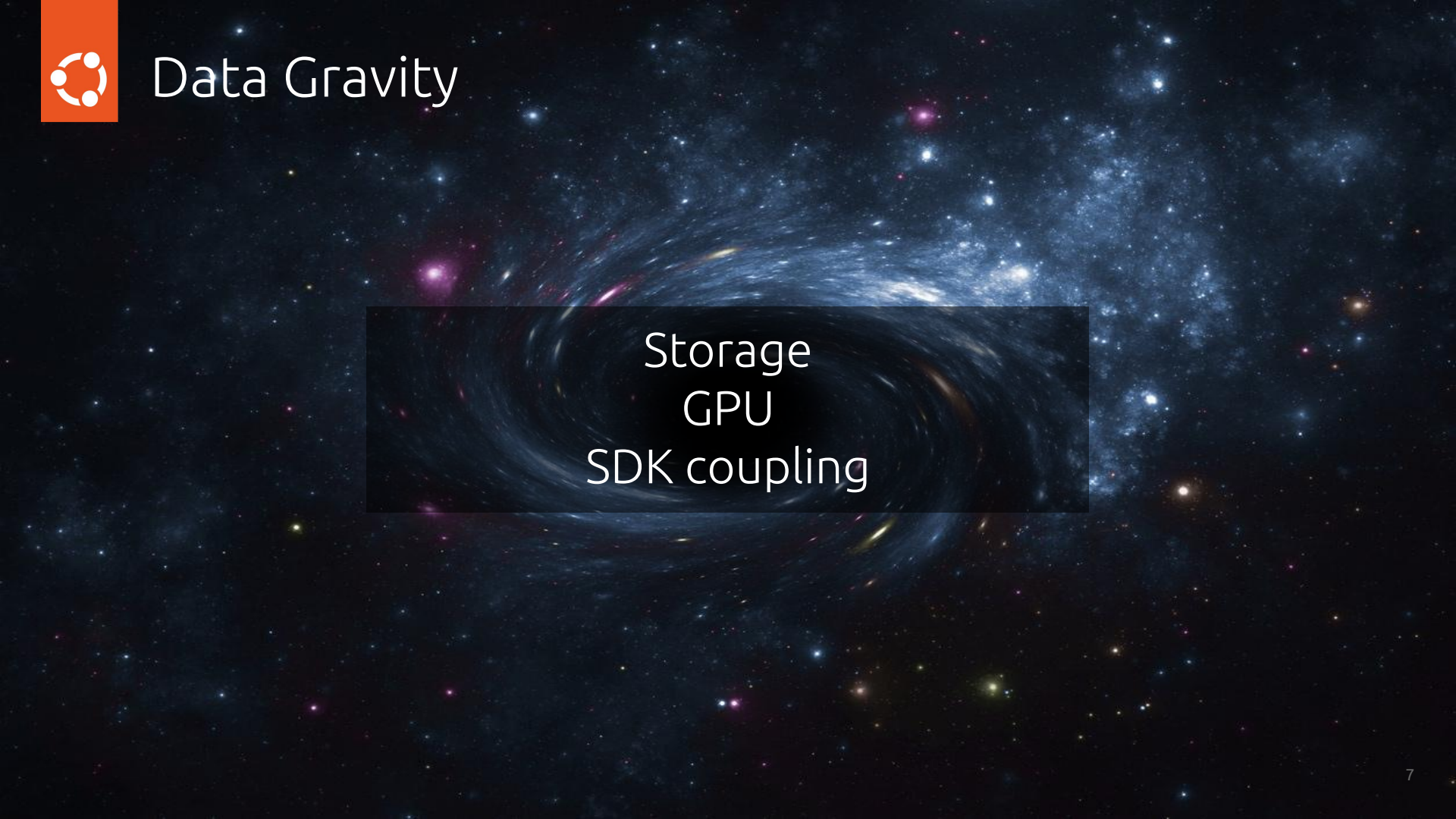
Training
Public Cloud (e.g. AWS)

Inference
Edge

Cloud / Infrastructure

Extend your environment anywhere

Data Gravity

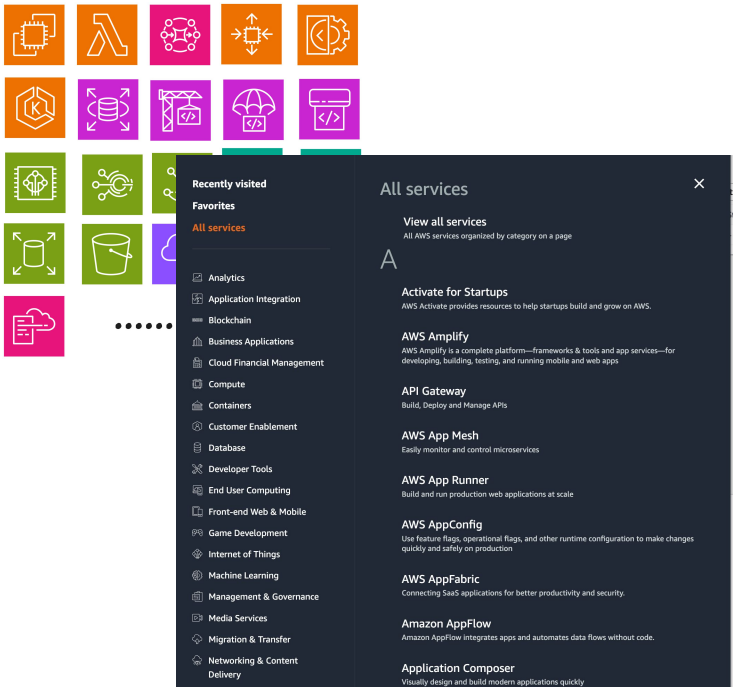


Cloud / Training

Reduce the complexity of your machine learning tools and operations

Training environment is **complex** to set up

A. Tons of clicks on GUI or API calls from CLI



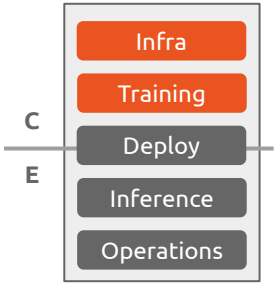
B. Complex IaC yaml files

```
resource "aws_subnet" "public_subnet" {
  vpc_id                  = aws_vpc.vpc.id
  count                   = var.subnet_count
  cidr_block              = cidrsubnet(var.cidr_vpc, var.cidr_network_bits, (count.index + length(sp
  availability_zone       = element(data.aws_availability_zones.all.names, count.index)
  map_public_ip_on_launch = true

  tags = {
    Name = "public-${element(data.a
  }

  depends_on = [aws_vpc.vpc]
}

resource "aws_internet_gateway" "in
  vpc_id     = aws_vpc.vpc.id
  depends_on = [aws_vpc.vpc]
}

resource "aws_eip" "nat_gateway_eip
  count      = var.subnet_count
  vpc        = true
  depends_on = [aws_internet_gatewa
}

resource "aws_nat_gateway" "nat_gat
  count         = 2
  allocation_id = aws_eip.nat_gatew
  subnet_id     = aws_subnet.public
  depends_on    = [aws_internet_gat
}
```

```
resource "aws_autoscaling_group" "my_sample_asg" {
  name                 = var.asg_name
  launch_configuration = aws_launch_configuration.my_sample_lc.name // Reference form above
  min_size             = 2
  desired_capacity     = 2
  max_size             = 4
  vpc_zone_identifier  = aws_subnet.private_subnet.*.id
  health_check_type    = "ELB"
  load_balancers       = [aws_elb.nginx_lb.name] // Add instances below Classic LB

  tag {
    key                 = "Name"
    value               = "asg-nginx-test"
    propagate_at_launch = true
  }

  lifecycle {
    create_before_destroy = true
  }
}

// LB security Group

resource "aws_security_group" "lb_sg" {
  name        = "${var.sg_name}-lb"
  description = "Managed by Terraform"
  vpc_id      = aws_vpc.vpc.id

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

Solution architecture

Open-source (cloud)



Public Cloud tools

Databases

MLOps (with Canonical Kubeflow)

Kubernetes (with Canonical MicroK8s)

Any laptop, VM, or server

AI model

Public Cloud agent

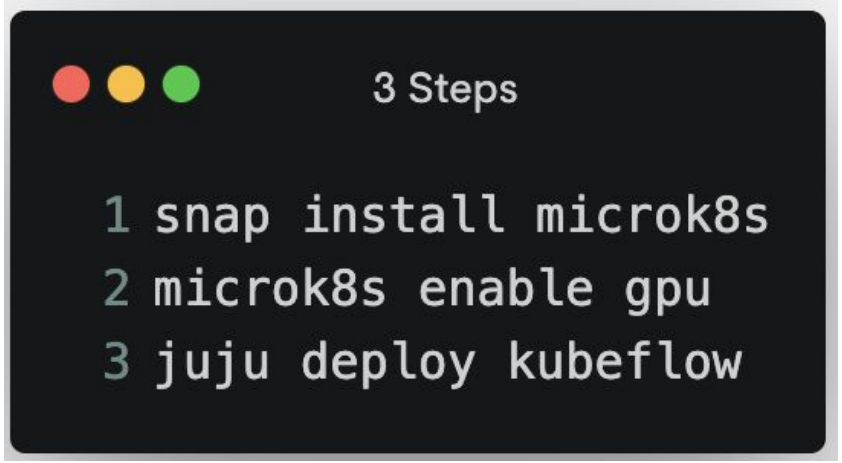
Container

Cameras

Training
Anywhere

Inference
Edge

Simplify your training environment - in 3 steps

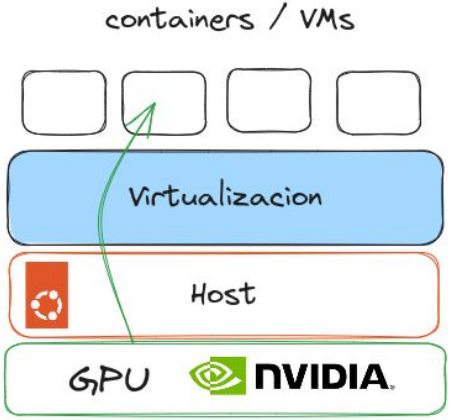


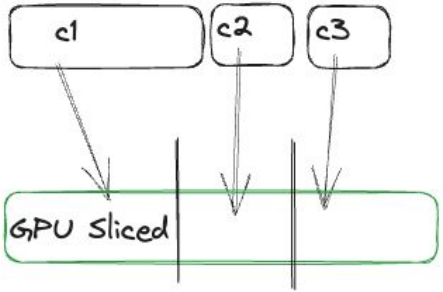
1. Set up a cloud agnostic environment
In one line

```
ubuntu@blue:~$ sudo snap install microk8s --channel 1.28/stable --classic
microk8s (1.28/stable) v1.28.7 from Canonical✓ installed
ubuntu@blue:~$
```

2. GPU Drivers and Scheduling In one line

```
ubuntu@blue:~$ sudo microk8s enable gpu
Infer repository core for addon gpu
Enabling NVIDIA GPU
Addon core/dns is already enabled
Addon core/helm3 is already enabled
Checking if NVIDIA driver is already installed
GPU 0: NVIDIA GeForce GTX 1660 Ti (UUID: GPU-15d01162-
Using host GPU driver
"nvidia" has been added to your repositories
Hang tight while we grab the latest from your chart re
...Successfully got an update from the "nvidia" chart
Update Complete. ⎈Happy Helming!⎈
NAME: gpu-operator
LAST DEPLOYED: Mon Mar 11 03:23:35 2024
NAMESPACE: gpu-operator-resources
STATUS: deployed
REVISION: 1
TEST SUITE: None
NVIDIA is enabled
ubuntu@blue:~$
```





3. MLOps E2E environment
In two lines

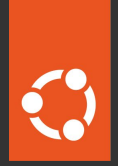
```
ubuntu@blue:~$ juju bootstrap microk8s
ubuntu@blue:~$ juju deploy kubeflow --channel 1.7/stable --trust
```

Experimenting

Training Pipelines

Model Versioning

Model Serving



Cloud to Edge / Deploy

Scale your solution with ease across your fleet of devices

Deploying a model is a multi-step process

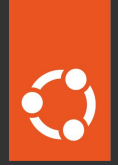
```
# ===== 0. Register
camera_1 $ sudo apt install default-jre
camera_1 $ export AWS_ACCESS_KEY_ID=<AWS_ACCESS_KEY_ID>
camera_1 $ export AWS_SECRET_ACCESS_KEY=<AWS_SECRET_ACCESS_KEY>
camera_1 $ export AWS_SESSION_TOKEN=<AWS_SESSION_TOKEN>
camera_1 $ curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip > greengrass-nucleus-latest.zip && unzip greengrass-nucleus-latest.zip -d GreengrassInstaller
camera_1 $ sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar ./GreengrassInstaller/lib/Greengrass.jar --aws-region us-east-2 --thing-name GreengrassQuickStartCore-18e2ce04721 --thing-group-name GreengrassQuickStartGroup --component-default-user ggc_user:ggc_group --provision true --setup-system-service true --deploy-dev-tools true
[..]
camera_999 $ [..]

# ===== 1. Publish
develop $ push repository/my_app:1.0.1

# ===== 2. Rollout
camera_1 $ pull repository/my_app:1.0.1
camera_1 $ rollout 1.0.1
[..]
camera_999 $ [..]
# [breaks]

# ===== 3. Rollback
camera_1 $ rollout 1.0.0
camera_2 $ rollout 1.0.0
[..]
camera_999 $ [..]
```

Deploying a model needs to be scalable

One line publish

```
develop $ upload --release=1.0/stable my_app
```

Transactional updates in case of failure



Edge / Inference

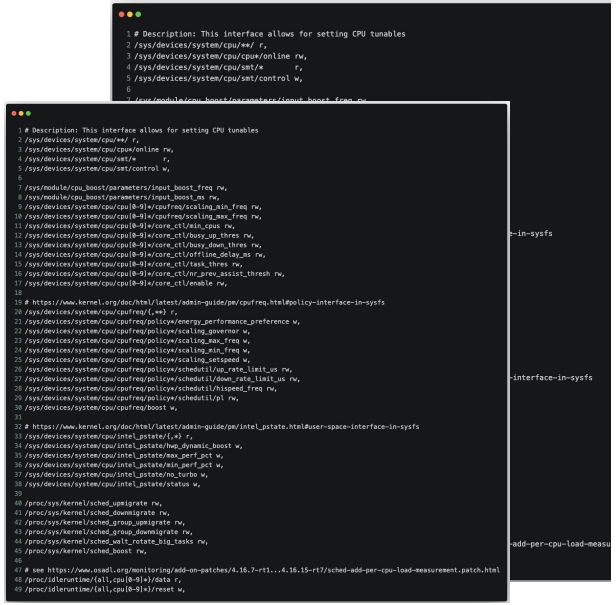
Security outside your datacenter

Traditional access to hardware

A. Unsecure full access

```
docker run \
  --priviledged \
  --ipc=host \
  --device=/dev/video0 \
  -v <host-path>:<container-path>
```

B. Complex granular security rules

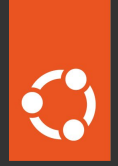


But.. is that how your phone works?

Secure and simple IoT

Only 2 lines

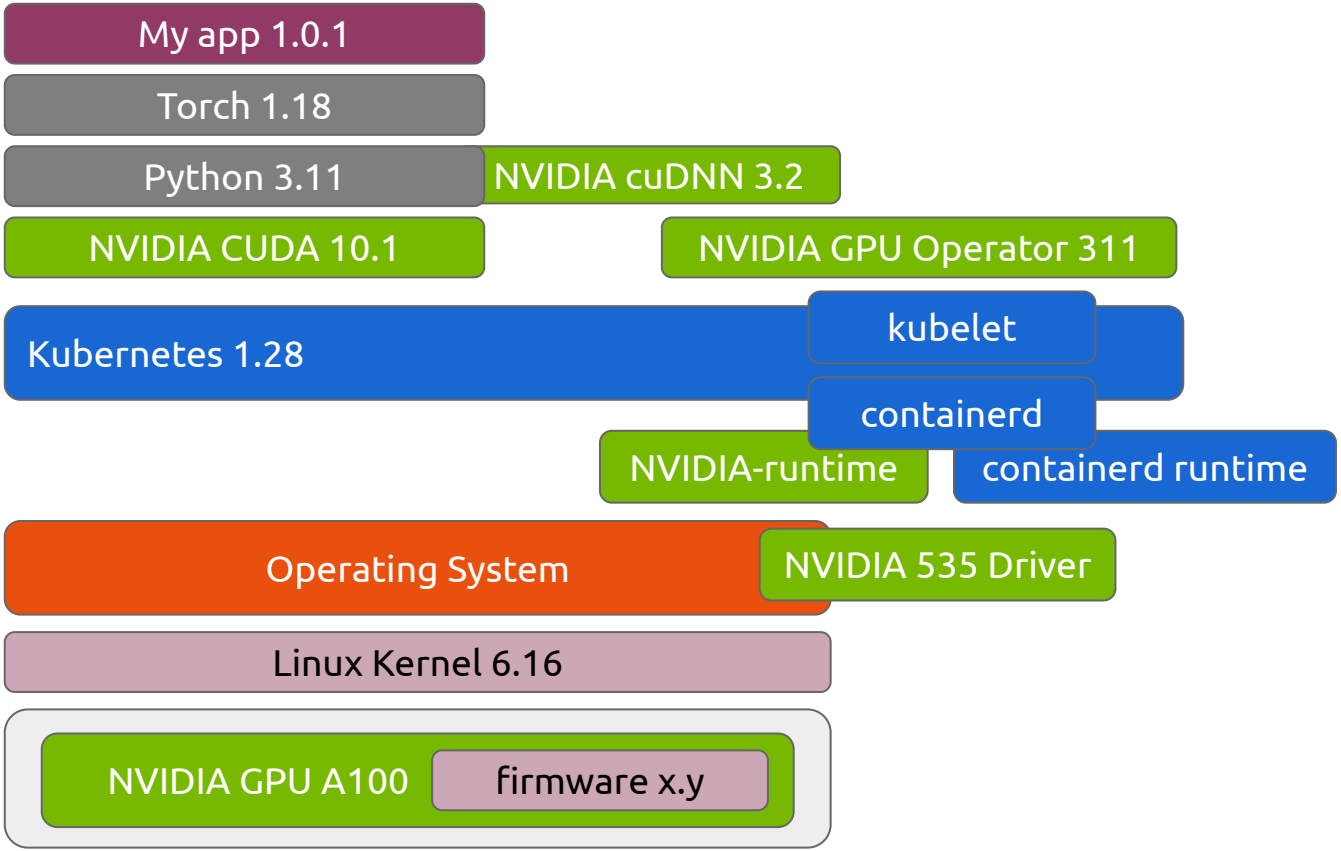
```
webcam-daemon:
  command: bin/python $SNAP/webcam.py
  daemon: simple
  plugs:
    - camera
    - opengl
```



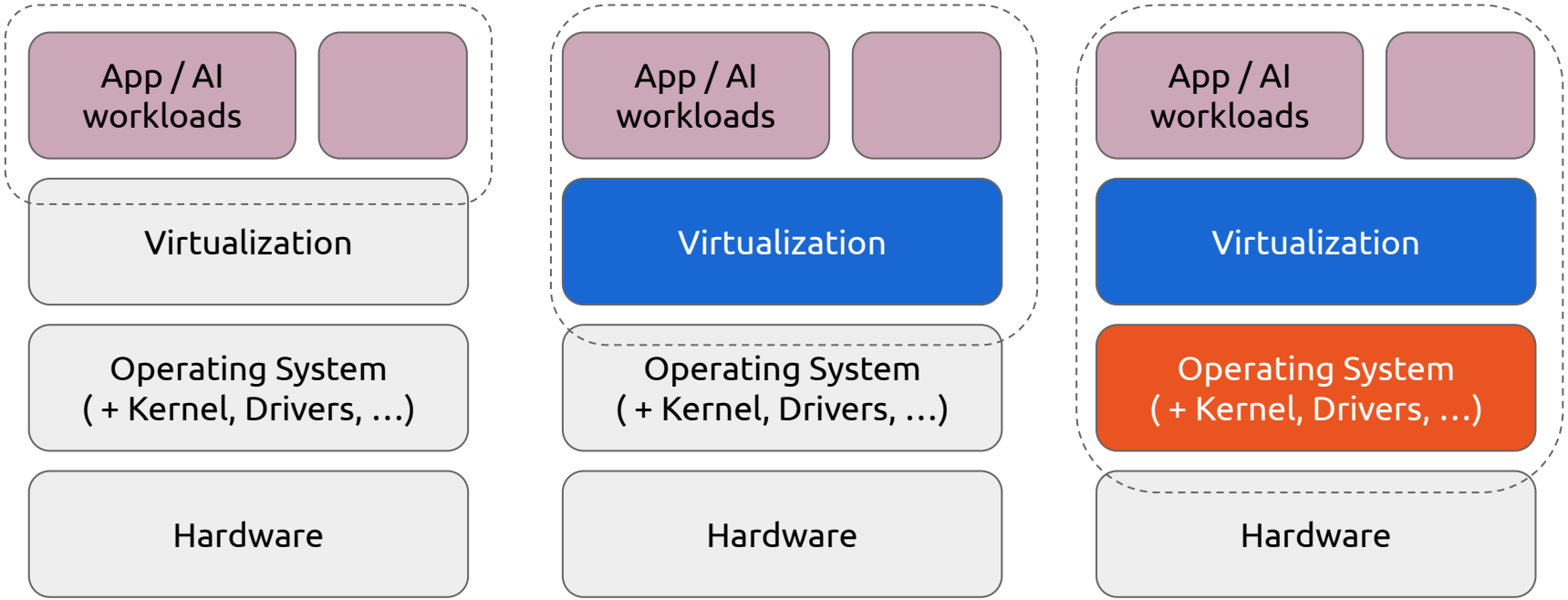
Edge / Operations

Robustness out in the wild

The AI Stack: more than just the model

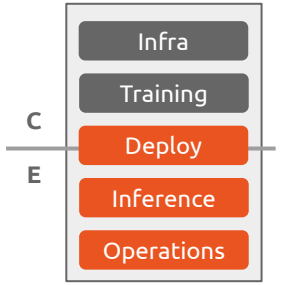


Updates and patches across all layers



Solution architecture

Open-source (edge)



Snapcraft

Databases

MLOps (with Canonical Kubeflow)

Kubernetes (with Canonical MicroK8s)

Any laptop, VM, or server

AI model

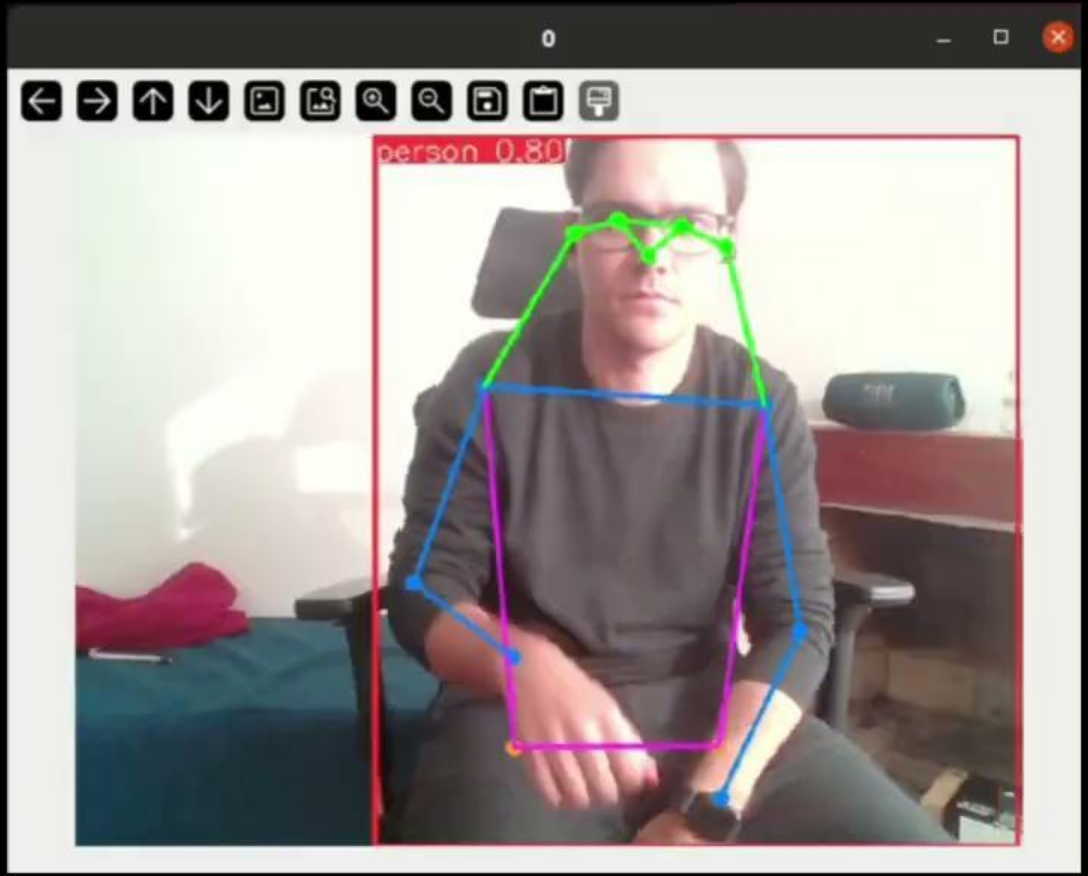
Snap

Container

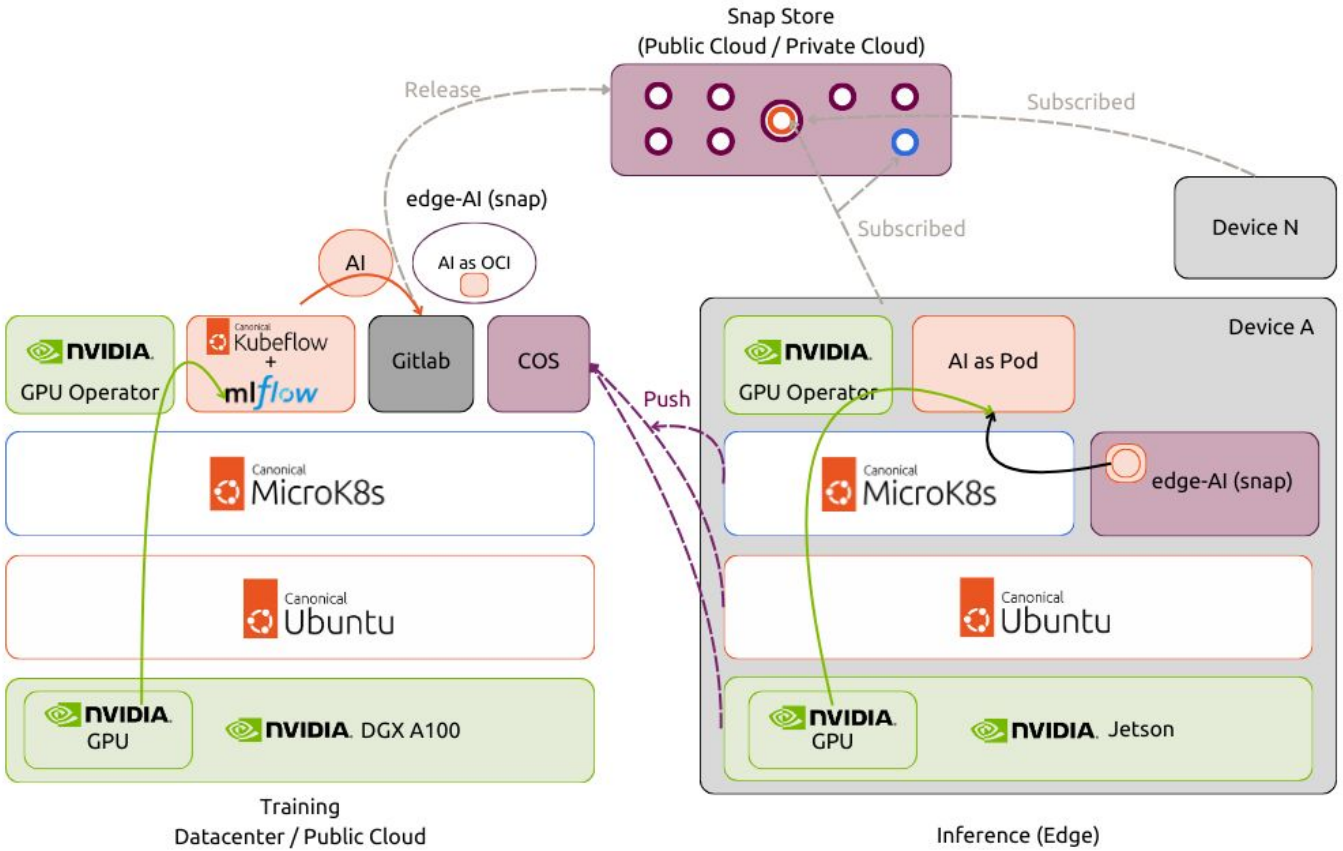
Cameras

Training
Anywhere

Inference
Edge

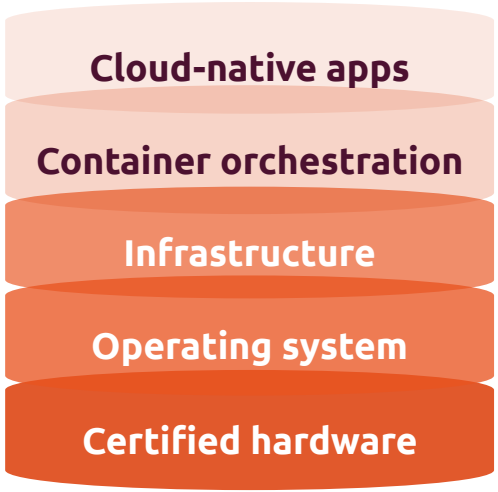

0
person 0.80

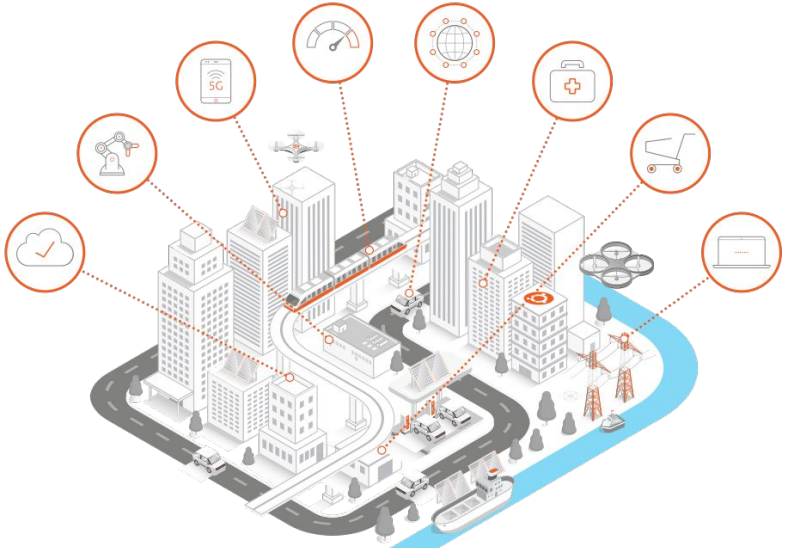
Open-source architecture



An end-to-end solution

For scalable smart cities





NVIDIA x Canonical

Meet with us at booth

1601

Let’s connect

LinkedIn

Joint webinar on April 4th

Ubuntu on Jetson: A Journey From Development to Production

Canonical

20 Years of innovation