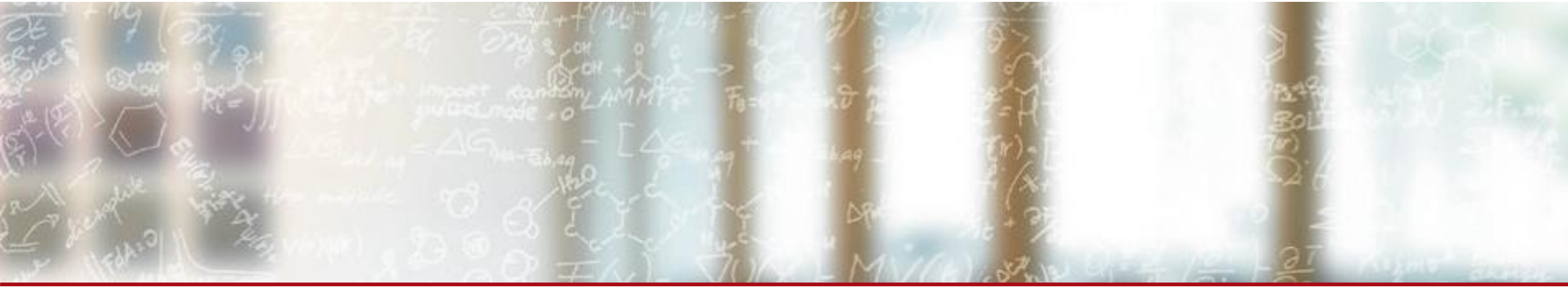
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



# A Cloud-Native Control Plane for Infrastructure and Platform Management - POC

## One control plane to manage them all, one YAML to bind them

Dino Conciatore – System Engineer

November 5th, 2024

HEPiX Fall – University of Oklahoma

# POC Goals

- Software defined Infrastructure
- Single manifest representing the desired state of the infrastructure and platform
- Portability
- Use the right tool for the right purpose
  - Infrastructure via Terraform
  - Node configuration via Ansible
  - API Interaction with API Helper (HTTP)
  - Git repository with GitLab Helper
  - …
  - …

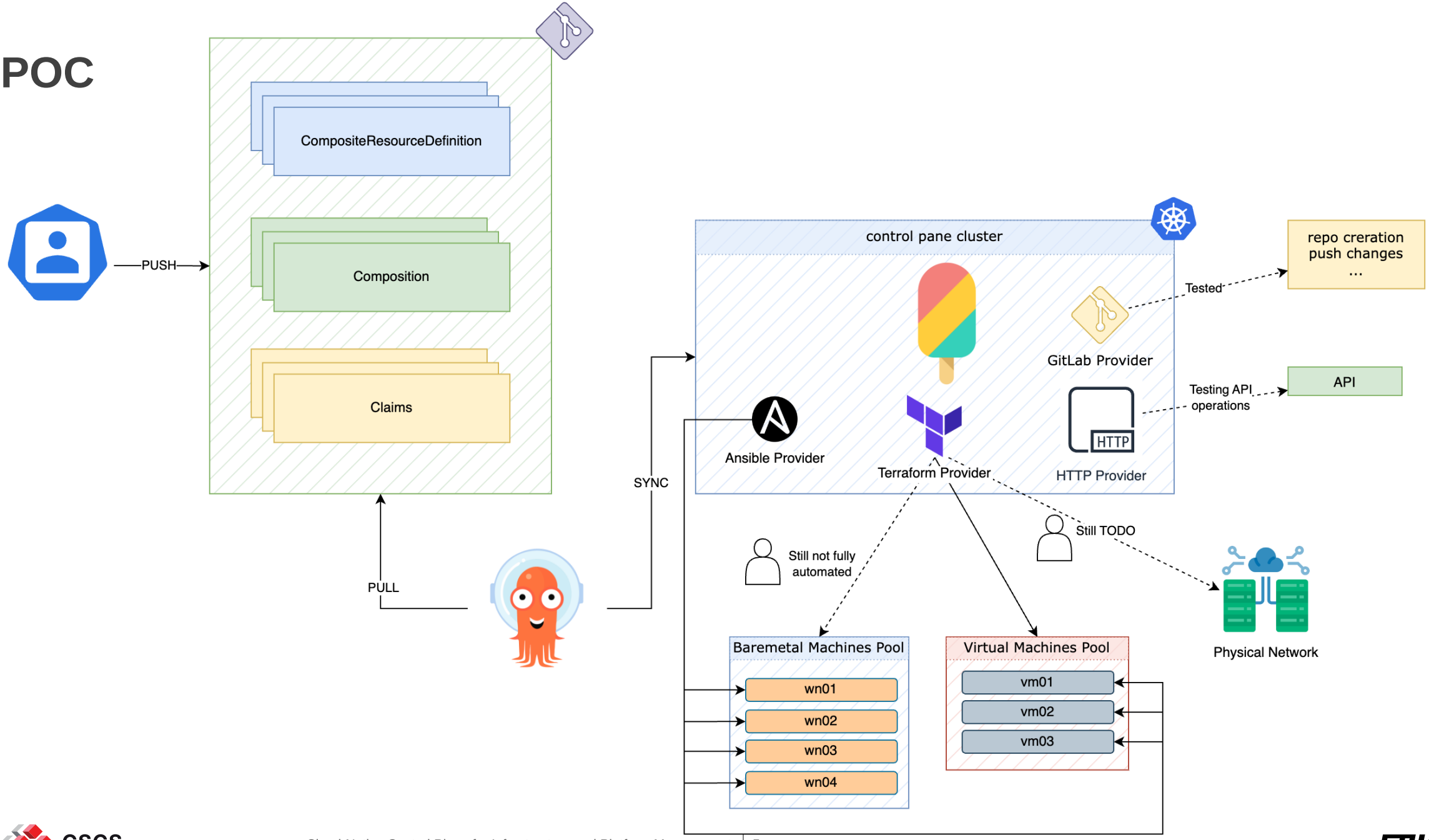
# Crossplane

- **Open-Source Control Plane**
  - Manage cloud resources and services
- **Kubernetes Native**
  - Extends Kubernetes with cloud-native capabilities
- **Infrastructure as Code**
  - Define and manage infrastructure using Kubernetes API and CRDs
- **Multi-Cloud Support**
  - Compatible with AWS, Azure, and more
- **Declarative API**
  - Use Kubernetes manifests to describe infrastructure
- **Composable Infrastructure**
  - Modular and reusable infrastructure components
- **Secure and Consistent**
  - Enforces policies and ensures consistent configurations
- **Scalable and Extensible**
  - Easily scales and integrates with other tools

# Componets required for infrastructure definition

- **Crossplane Composition**
  - Crossplane Composition is a method of defining cloud infrastructure in a reusable and modular way
- **CompositeResourceDefinition**
  - Crossplane defines a custom resource that bundles multiple resources into a single, reusable unit, enabling declarative infrastructure management.
- **Claim**
  - A Claim in Crossplane is a high-level request for a managed resource, letting developers provision infrastructure without cloud-specific details.

# POC


CompositeResourceDefinition
Composition
Claims
PUSH
PULL
SYNC
control pane cluster
Ansible Provider
Terraform Provider
GitLab Provider
HTTP
HTTP Provider
Tested
repo creration
push changes
...
Testing API
operations
API
Still not fully
automated
Still TODO
Physical Network
Baremetal Machines Pool
wn01
wn02
wn03
wn04
Virtual Machines Pool
vm01
vm02
vm03

# Benefits

- **Modularity:** Create reusable infrastructure components.
- **Abstraction:** Simplifies complex infrastructure management.
  - **Portability**
- **Declarative:** Uses Kubernetes CRDs for infrastructure as code.

# Ansible provider configuration

- ssh keys
  - Key used to connect to the hosts
  - Keys are stored in Vault and exposed to the cluster with external-secrets
- Ansible requirements
  - roles
  - collections

# Terraform provider configuration

- Backend
  - S3
- Required provider configurations
  - Credentials to connect

# Crossplane Custom resource definition

- Defines the varibles template in Open API v3 Schema format

```
schema:
  openAPIV3Schema:
    type: object
    properties:
      spec:
        type: object
        properties:
          harvester_image:
            type: string
          clustername:
            type: string
          cpus:
            type: string
```

# Crossplane Compositions

- Terraform module to create VMS
- Ansible to create a simple Slurm cluster

# Claim example

```yaml
---
apiVersion: xplank.io/v1alpha1
kind: VMCluster
metadata:
  name: hepix1
spec:
  harvester_image: "ubuntu-jammy"
  clustername: "hepix1"
  node_count: "4"
  cpus: "8"
  memory: 8Gi
  disk_size: 20Gi
  vlan: "83"
```

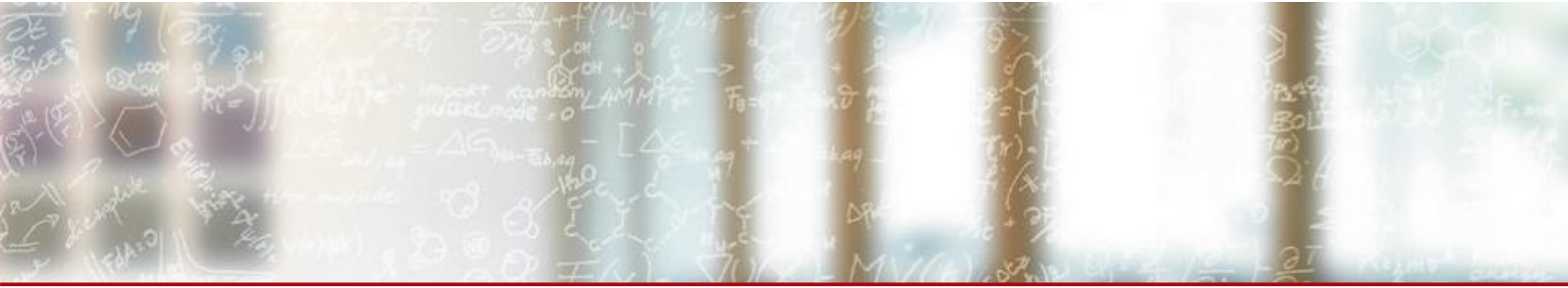
# Ansible at scale - dosen’t scale

Ansible at scale is really crap

- Dynamic *delegate_to*
  - testing
- Async
  - Helps but is not enough
- ansible-pull
  - Similar approach to Puppet

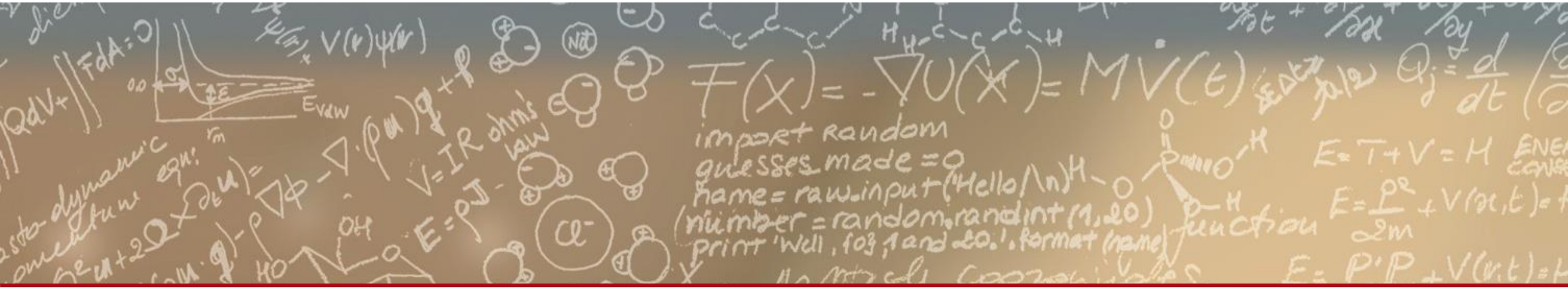
Open discussion:

- Which configuration tool are you using at scale?
- Wich approach?

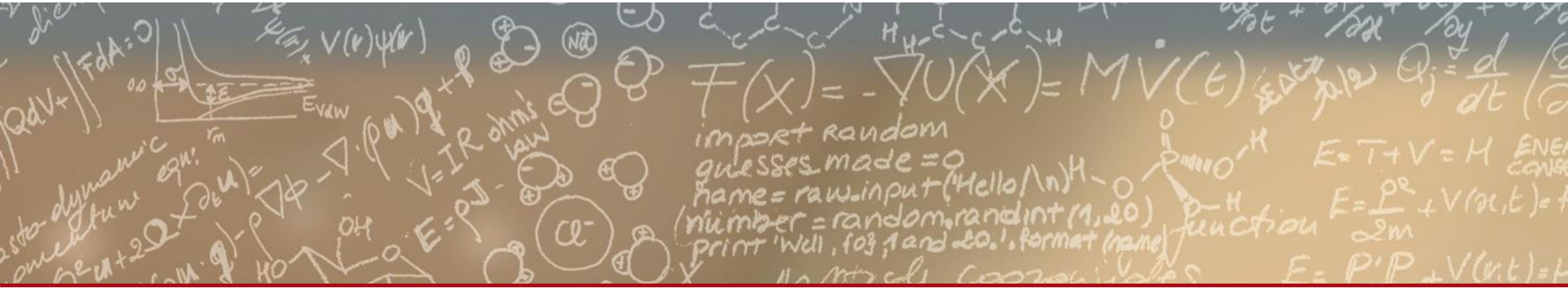


# DEMO TIME

Creation of a Slurm cluster on top of some VMs hosted on Harvester


import random
guesses_made = 0
name = raw_input('Hello! What is your name?\n')
number = random.randint(1, 20)
print 'Well, ' + name + ', I am thinking of a number between 1 and 20.'

# Questions?




# Thank you for your attention.