MINDSONAI

THE MLOPS FRAMEWORK

ZENML

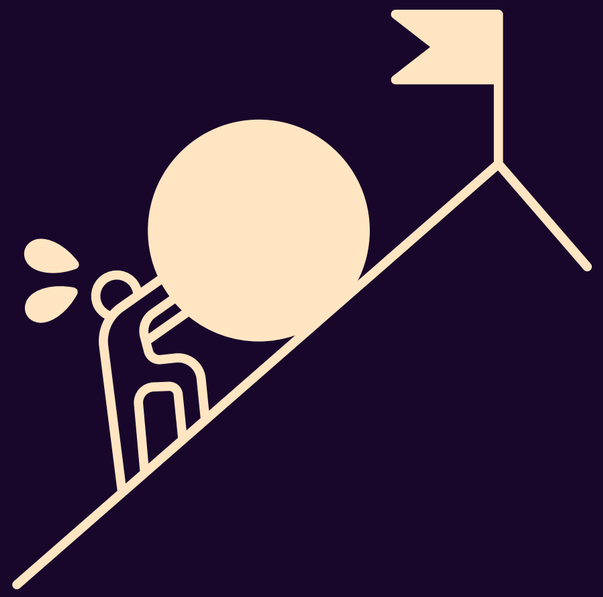
https://linkedin.com/rishabhio

JIO 2024

Why Productionalizing a model is a hassle ?

Reproducibility

Model Versioning

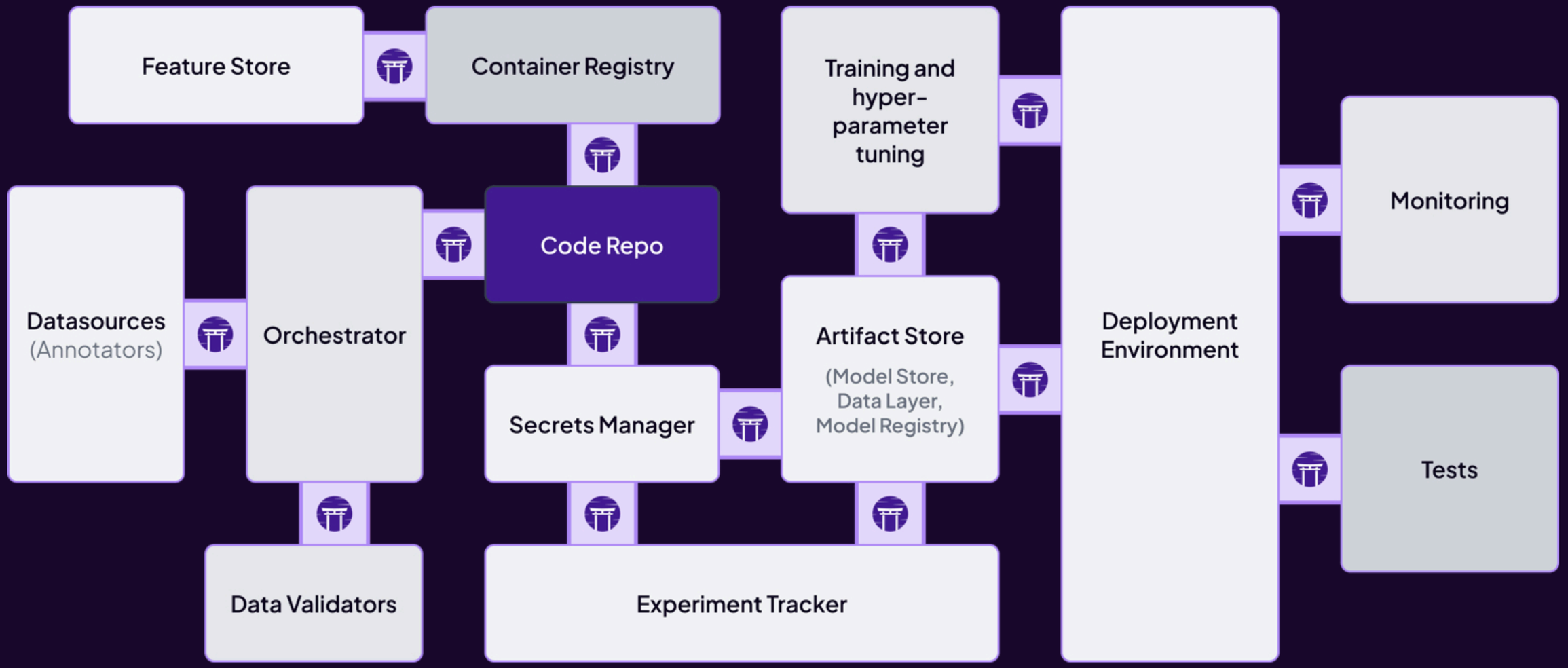


Scalibility

Re-Training

Monitoring

And 10s of other tasks



There are always many components to deal with.

Now What's an even bigger Challenge?

Dev Phase

Let's get done with the model training on Jupyter Lab first.

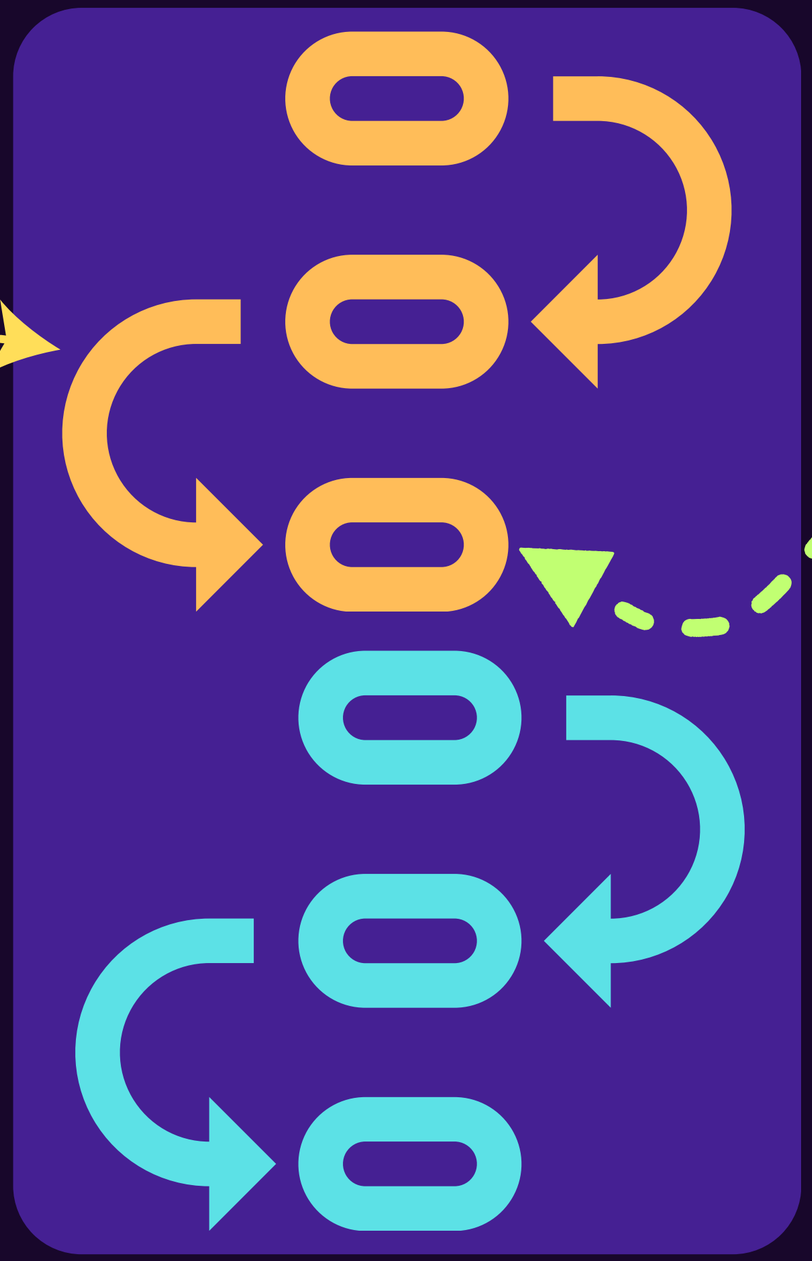
ML-OPS Practices

But there's a solution to every problem!

Introducing 'ZenML'

ZenML is an open-source ML-Ops Framework which serves as a solution to many of the common ML-OPs problems.

'ZenML'



Pipelines

Steps

Stacks

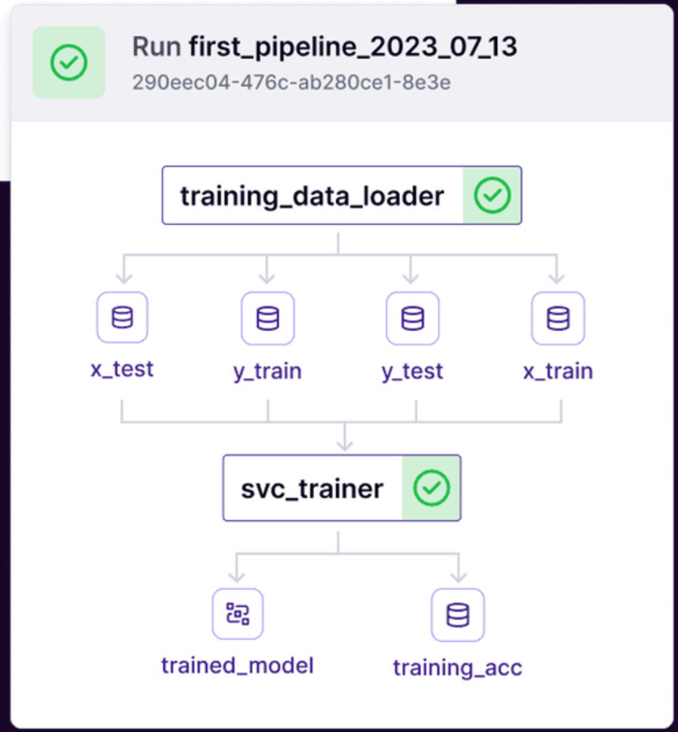
In the quest for production-ready ML models, workflows can quickly become complex

Decoupling and standardizing stages such as data ingestion, preprocessing, and model evaluation allows for more manageable, reusable, and scalable processes

ZenML

```python
@pipeline
def first_pipeline(gamma: float = 0.002):
    X_train, X_test, y_train, y_test = training_data_loader()
    svc_trainer(gamma=gamma, X_train=X_train, y_train=y_train)

if __name__ == "__main__":
    first_pipeline(gamma=0.0015)
```



The Best Part:

ZenML Pipelines are nothing but simple Python Code.

Let's Get Coding

In the next steps, let's create a simple Pipeline using ZenML

Step-1 Create Venv

```
python -m venv zen_env
```

Step-2 Install ZenML

```
pip install "zenml[server]"
```

Step-3 Verify Install

```
zenml version
```

Step-4 File

Create a new File in VsCode: run.py

Step-5: Imports

```
from zenml import pipeline, step
```

Step-6: Define a Step

The magic starts with these decorators.

```
@step
def load_data() -> dict:
    """Simulates loading of training data and labels."""
    “”” Please replace it with the actual logic based on your
        project”””

    training_data = [[1, 2], [3, 4], [5, 6]]
    labels = [0, 1, 0]

    return {'features': training_data, 'labels': labels}
```

Step-7: Another Step

The magic starts with these decorators.

```
@step
def train_model(data: dict) -> None:
    """
    A mock 'training' process that also demonstrates using the input data.
    In a real-world scenario, this would be replaced with actual model
fitting logic.
    """
    total_features = sum(map(sum, data['features']))
    total_labels = sum(data['labels'])

    print(f"Trained model using {len(data['features'])} data points. "
          f"Feature sum is {total_features}, label sum is {total_labels}")
```

Step-8: Pipeline

And the magic continues

```
@pipeline
def simple_ml_pipeline():
    """Define a pipeline that connects the steps."""
    dataset = load_data()
    train_model(dataset)
```

Step-9: Run Object

run -> executes the pipeline

```
if __name__ == "__main__":
    run = simple_ml_pipeline()
    # You can now use the `run` object to see steps, outputs, etc.
```

@step

is a decorator that converts its function into a step that can be used within a pipeline

@pipeline

defines a function as a pipeline and within this function, the steps are called and their outputs link them together

Step-10: Execute

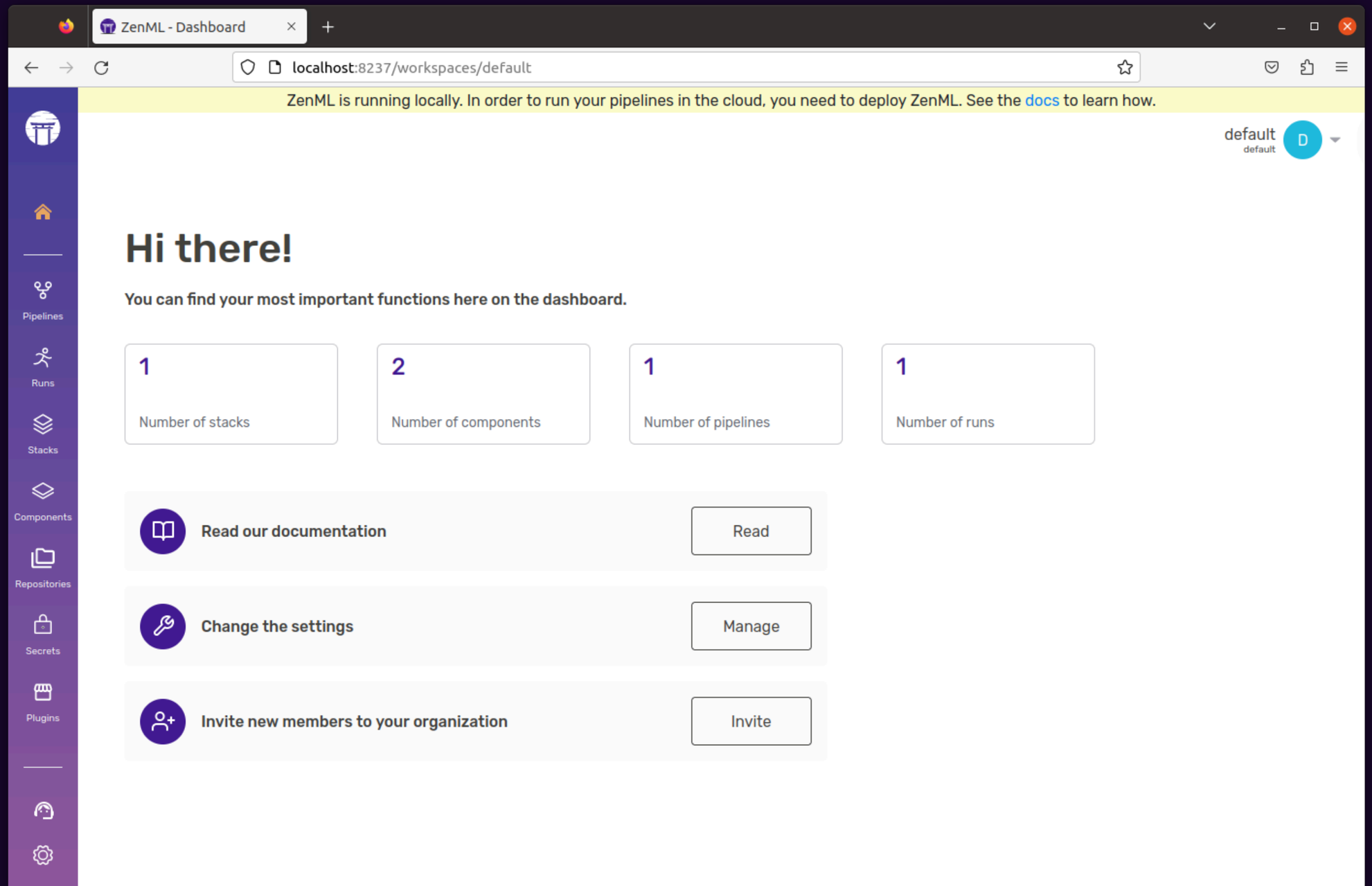
```
$ python run.py
```

```
Initiating a new run for the pipeline: simple_ml_pipeline.
Registered new version: (version 2).
Executing a new run.
Using user: hamza@zenml.io
Using stack: default
  orchestrator: default
  artifact_store: default
Step load_data has started.
Step load_data has finished in 0.385s.
Step train_model has started.
Trained model using 3 data points. Feature sum is 21, label sum is 1
Step train_model has finished in 0.265s.
Run simple_ml_pipeline-2023_11_23-10_51_59_657489 has finished in 1.612s.
Pipeline visualization can be seen in the ZenML Dashboard. Run zenml up to
see your pipeline!
```

Step-11: Explore

Explore the Dashboard

```
$ zenml up
```


ZenML - Dashboard
localhost:8237/workspaces/default
ZenML is running locally. In order to run your pipelines in the cloud, you need to deploy ZenML. See the docs to learn how.
default
default
D
Pipelines
Runs
Stacks
Components
Repositories
Secrets
Plugins
Hi there!
You can find your most important functions here on the dashboard.
1
Number of stacks
2
Number of components
1
Number of pipelines
1
Number of runs
Read our documentation
Read
Change the settings
Manage
Invite new members to your organization
Invite

Visit Dashboard

http://127.0.0.1:8237

Username: default

Password: Keep it Blank

If you have closed the browser tab with the ZenML dashboard, you can always reopen it by running:

```
zenml show
```

in your terminal.

Shut down the Server

```
$ zenml down
```

Side Note:

Add New VirtualEnv in your JupyterLab

```
pip install ipython

pip install ipykernel

ipython kernel install --user --name=env_zen

python -m ipykernel install --user --name=env_zen\n
```