Deployment Guide

Step-by-step guide to deploying a new version of Vector Inference on Vector clusters

Development Setup

Prerequisites

- Python 3.10 or newer
- uv for dependency management

Setting Up Development Environment

1. Clone the repository:

```
git clone https://github.com/VectorInstitute/vector-inference.git
cd vector-inference
```

2. Install development dependencies:

```
uv sync --all-extras --group dev
```

3. Install pre-commit hooks:

```
pre-commit install
```

NOTE If you prefer using virtual environments, step 2 creates a virtual environment by default, run the following to activate:

```
source .venv/bin/activate
```

Development Workflow

Development Workflow

1. Branch off of `main`, name your branch with the following convention: `f/$FEATURE_NAME`, `b/$BUG_FIX` or `r/RELEASE_TAG`.
2. Make new changes in your branch, and add/modify relevant unit tests in `tests`. Please follow the styling guide in the next section.
3. Test your changes, once ready, open a PR for review. See below for detailed testing guide.
4. The PR will trigger pre-commit hooks and run GitHub actions that would check code style and linting, run unit tests, build documentation and Docker images. All GitHub actions need to pass (along with 2 approvements) to merge the PR.

Code Style and Linting

We use several tools to ensure code quality:

- **ruff** for linting and formatting
- **mypy** for type checking

You can run these tools with:

```
# Linting
uv run ruff check .

# Type checking
uv run mypy

# Format code
uv run ruff format .
```

!!! note "Pre-commit Hooks" The pre-commit hooks will automatically run these checks before each commit. If the hooks fail, you will need to fix the issues before you can commit.

Testing

Testing new changes involves 2 major steps: unit tests and integration tests.

Unit Tests

All new features and bug fixes should include tests. We use pytest for testing:

```
# Change to test directory
cd tests

# Run all tests
```

```
uv run pytest


# Run tests with coverage
uv run pytest --cov=vec_inf
```

Integration Tests

Integration tests are stored in a different repository: vec-inf-maintenance

1. Clone the repository and run `cd vec-inf-maintenance/deployment`
2. Follow the README to test model launch and inference

Release Guide

Release PR

It is recommended to create a dedicated branch and PR for a new release:

1. Update version in `pyproject.toml`.
2. Run `uv lock --upgrade` to update the lock file.
3. Pull new images on the cluster if inference engine version is updated, more details in next section.
4. Update model tracking information and configs accordingly, more details below.

Update images

If the inference engine version is updated, the Docker GitHub action will build new images tagged by the inference engine version and pushed to DockerHub:

1. Go to `vec-inf-maintenance/maintenance`.
2. Run `pull_image.sh $ENGINE_NAME $VERSION`, e.g. `pull_image.sh vllm 0.14.0`. This will pull the new image to `/model-weights/vec-inf-shared`, and update the symlink accordingly.
3. Go to `vec-inf-maintenance/deployment`.
4. Run model launch and inference tests to ensure the new images are working properly.

Update model tracking information and configs

For any new release, always sync the latest cached configs (`models.yaml` and `environment.yaml`) on Killarney cluster (`/model-weights/vec-inf-shared`) to the repository:

- If new models were downloaded and/or supported on the cluster, they should be reflected in `MODEL_TRACKING.md`.
- Whenever changes are made to the cached configs on any cluster, the cached configs should be pushed to `vec-inf-maintenance/vec-inf-shared/$CLUSTER_NAME`, e.g. `vec-inf-maintenance/vec-inf-shared/bon-echo`

Create a new release on GitHub

1. Create a new tag and release on GitHub
2. Add description for the new release
3. Documentation for the new version will be automatically deployed

Maintenance

Cached Configs and Apptainer Images

In the `/model-weights` folder, the following files exists:

- `environment.yaml`: Cached environment config
- `models.yaml`: OLD cached model config. This is from pre-v0.8.0, it is no longer being updated, cached model config path was determined directly from the package instead of read from cached environment config. This is kept for backward compatibility.
- `models_v0.8.0.yaml`: NEW cached model config. This is the real "models.yaml" being used by latest versions after v0.8.0. Consider removing the old cached model config and rename this file as `models.yaml` in the near future.
- `vector-inference_x.y.z.sif`: OLD Vector Inference image, this is from pre-v0.8.0, where only vLLM is baked in and the tag is based on vLLM version, discard in the future.
- `vector-inference_latest.sif`: OLD Vector Inference image symlink, this is from pre-v0.8.0 pointing to `vector-inference_x.y.z.sif`, discard in the future.
- `vector-inference-vllm_x.y.z.sif`: Vector Inference image with vLLM baked in, tagged by vLLM version number. Remove old ones as see fit.
- `vector-inference-sglang_x.y.z.sif`: Vector Inference image with SGLang baked in, tagged by SGLang version number. Remove old ones as see fit.
- `vector-inference-vllm_latest.sif`: Vector Inference vLLM image symlink, always point to the latest working version.
- `vector-inference-sglang_latest.sif`: Vector Inference SGLang image symlink, always point to the latest working version.

Model weights tracking

Use `MODEL_TRACKING.md` in the `vec-inf-maintenance` repo to keep track of cached model weights.

- `MODEL_TRACKING.md`: Cached model weights list (i.e. downloaded weights in `/model-weights`), and whether or not it's currently supported by `vec-inf` (i.e. cached model config)