Create and manage jobs

Submitting a job

To submit your job to Anyscale, use the Python SDK or CLI and pass in any additional options or configurations for the job.

By default, Anyscale uses your workspace or cloud to provision a cluster to run your job. You can define a custom cluster through a compute config or specify an existing cluster.

Once submitted, Anyscale runs the job as specified in the entrypoint command, which is typically a Ray Job. If the run doesn't succeed, the job restarts using the same entrypoint up to the number of `max_retries`.

CLI **Python SDK**

```
anyscale job submit --name=my-job \
  --working-dir=. --max-retries=5 \
  --image-uri="anyscale/image/IMAGE_NAME:VERSION" \
  --compute-config=COMPUTE_CONFIG_NAME \
  -- python main.py
```

With the CLI, you can either specify an existing compute config with `--compute-config=COMPUTE_CONFIG_NAME` or define a new one in a job YAML.

For more information on submitting jobs with the CLI, see the reference docs.

TIP

For large-scale, compute-intensive jobs, avoid scheduling Ray tasks onto the head node because it manages cluster-level orchestration. To do that, set the CPU resource on the head node to 0 in your compute config.

Ask AI

Defining a job

With the CLI, you can define jobs in a YAML file and submit them by referencing the YAML:

```
anyscale job submit --config-file config.yaml
```

For an example of defining a job in a YAML, see the reference docs.

Waiting on a job

You can block CLI and SDK commands until a job enters a specified state. By default, `JobState.SUCCEEDED` is used. See all available states in the reference docs.

CLI Python SDK

```
anyscale job wait -n job-wait
```

When you submit a job, you can specify `--wait`, which waits for the job to succeed or exits if the job fails.

```
anyscale job submit -n job-wait --wait -- sleep 30
```

For more information on submitting jobs with the CLI, see the reference docs.

Terminating a job

You can terminate a job from the **Job** page or using the CLI/SDK:

CLI Python SDK

```
anyscale job terminate --id 'prodjob_...'
```

For more information on terminating jobs with the CLI, see the reference docs.

Archiving a job

Archiving jobs hide them from the job list page, but you can still access them through the CLI and SDK. The cluster associated with an archived job is archived automatically.

To be archived, jobs must be in a terminal state. You must have created the job or be an organization admin to archive the job.

You can archive jobs in Anyscale console or through the CLI/SDK:

CLI Python SDK

```
anyscale job archive --id 'prodjob_...'
```

For more information on archiving jobs with the CLI, see the reference docs.

Managing dependencies

When developing Anyscale jobs, you may need to include additional Python packages or system-level dependencies. There are several ways to manage these dependencies:

Using a requirements.txt file

The simplest way to manage Python package dependencies is by using a `requirements.txt` file.

1. Create a `requirements.txt` file in your project directory:

   ```
   emoji==2.12.1
   numpy==1.21.0
   ```

2. When submitting your job, include the `-r` or `--requirements` flag:

CLI **Python SDK**

```
anyscale job submit --config-file job.yaml -r ./requirements.txt
```

This method works well for straightforward Python package dependencies. Anyscale installs these packages in the job's environment before running your code.

Using a custom container

For more complex dependency management, including system-level packages or specific environment configurations, use a custom container:

1. Create a `Dockerfile`:

```
FROM anyscale/ray:2.10.0-py310

# Install system dependencies if needed
RUN apt-get update && apt-get install -y <your-system-packages>

# Install Python dependencies
COPY requirements.txt /tmp/
RUN pip install -r /tmp/requirements.txt
```

2. Build and submit the job with the custom container:

CLI **Python SDK**

```
anyscale job submit --config-file job.yaml --containerfile Dockerfile
```

This method gives you full control over the job's environment, allowing you to install both system-level and Python packages.

Using pre-built custom images

For frequently used environments, you can build and reuse custom images:

1. Build the image:

CLI Python SDK

```
anyscale image build -n my-custom-image --containerfile Dockerfile
```

2. Use the built image in your job submission:

CLI Python SDK

```
anyscale job submit --config-file job.yaml --image-uri anyscale/image/my-
custom-image:1
```

This approach is efficient for teams working on multiple jobs that share the same dependencies.