This is a formulation of the agent goal, entities, and environment around the '*stay-or-go*' task for scheduling HPC jobs.

→ HPCRL GOAL: For a job J*i*, HPCRL correctly chooses the local cluster or cloud cluster as a better platform to run that job. The objective function is **to minimize the regret from agent choice: difference between the decision made and the optimal decision.**
→ When is a platform better for a job run? When Total utility(choice platform) obtained from running that job is at least as good as Total Utility(alternative platform).
→ How is the Total Utility for a job on a platform estimated? Weight **sum** of **Number of seconds** it takes to run the job to completion plus **dollar cost** of the job run.

Cloudbursting environment?
→ State space
[Job submitted, Job failed, Job complete];
[CPU constraint + memory constraint + time spent on queue]
[local cluster available, local cluster unavailable];
Start state? Goal state? Other states? Not sure how to model this yet

Action space: if initial submission:
→ Run on cloud, Run local, wait
if resubmission:
→ Rerun on cloud, Rerun local, wait

For instance, we define a typical state, observation, action taken by agent, and resulting state.

State S_i: Job submitted + X CPU + Y memory + time spent in queue by job
Action A: Run on Cloud
State S_j: Job Failed + 20 dollars spent + 5 hours lost
Reward:

State S_i: Job submitted + X CPU + Y memory + time spent in queue by job
Action B: Wait for space on local cluster
State S_j: local cluster available
Reward:

Open questions:
Is there any mechanism to show Beocat users how long their jobs might wait before the cluster is freed up to run their jobs? Assumption is that this wait time is negligible for the cloud option.
How do we model waiting jobs? How important is time lost while jobs are waiting to be scheduled?