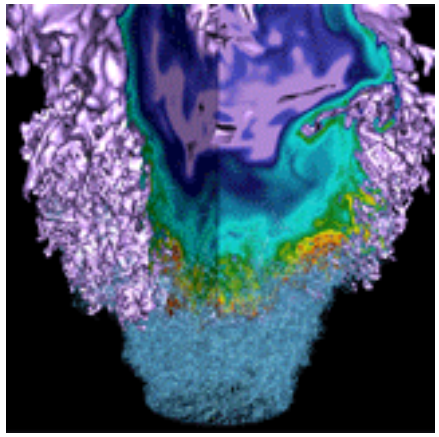
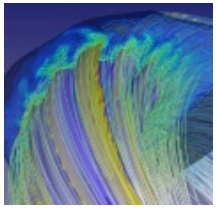
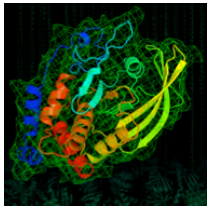
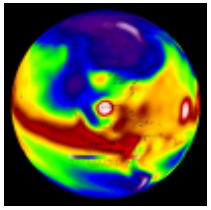
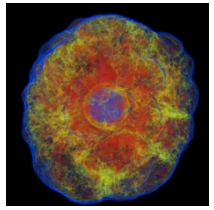
Image Gateway











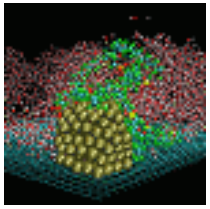
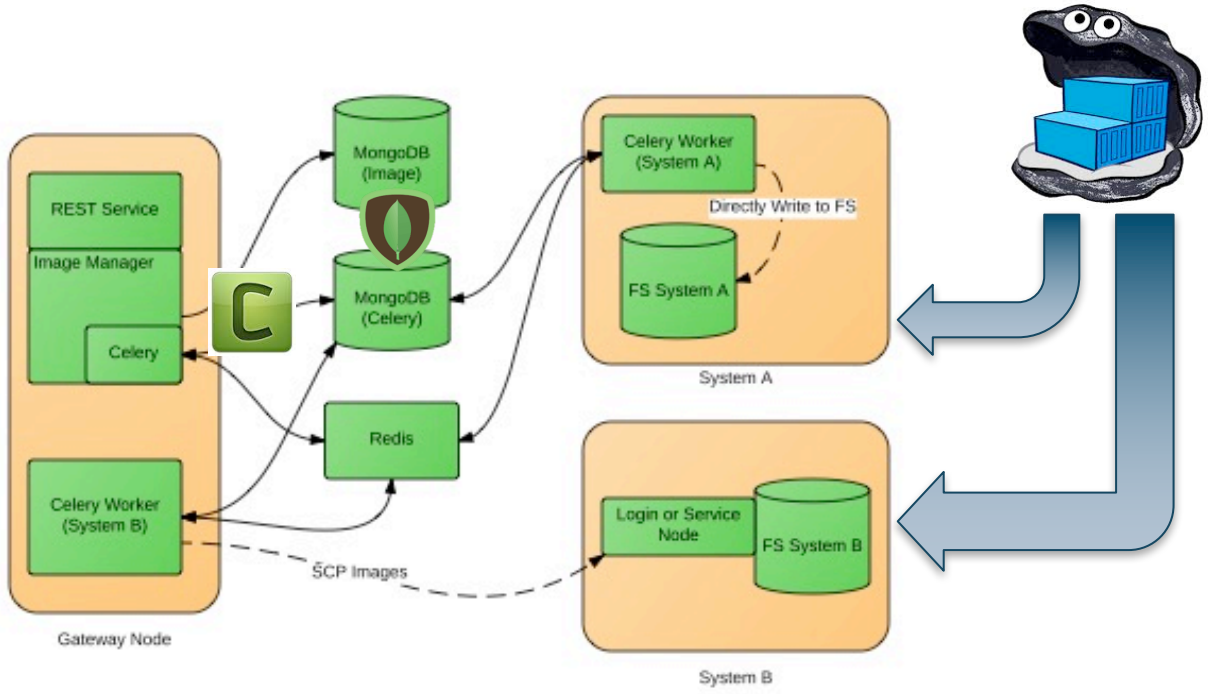


Image Gateway Design



- Python Flask Application provides REST interface
- Mongo Database stores image metadata and provides an index of available images
- Python Celery provides a distributed queueing system
- Celery "Workers" do the actual image manipulation including pulling Docker Images from DockerHub or Registries

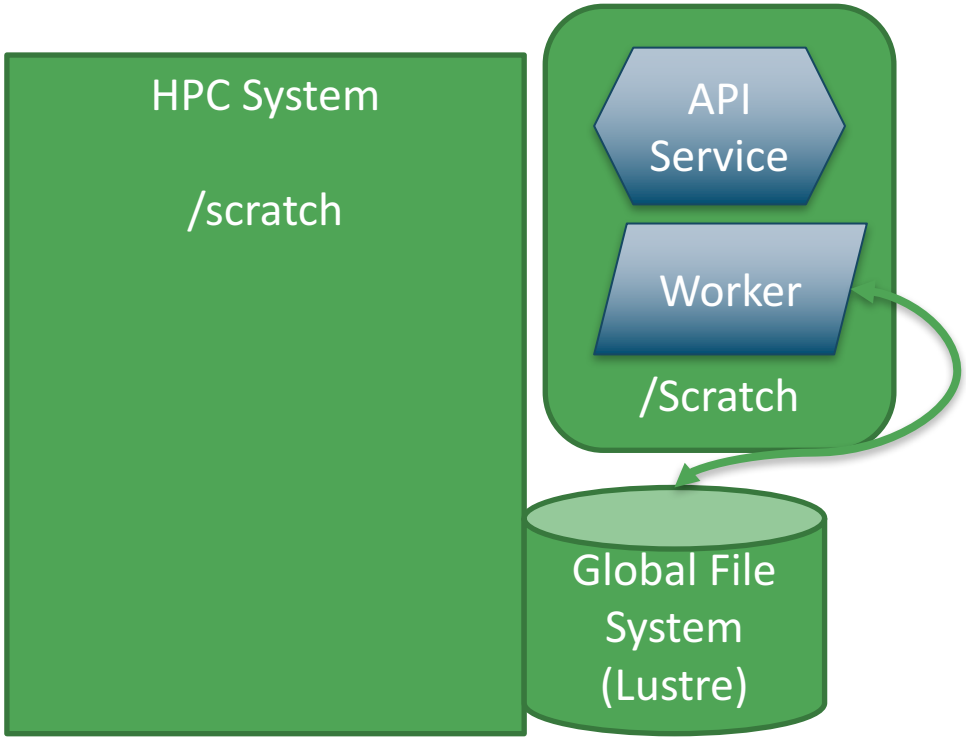
Image Gateway Options and Considerations

- Image Gateway can be run with the API service or remote
- Image Workers can write image directly (local) or copy them via SCP
- Image Gateway can support multiple systems
- Worker should have access to any external registries (e.g. DockerHub)
- Workers should have sufficient local disk or ramdisk for caching layers and unpacking images

Deployment Options

1. API service and worker(s) colocated with local access
2. API service and worker(s) colocated with ssh access to global file system
3. API service separate from worker with worker having local access

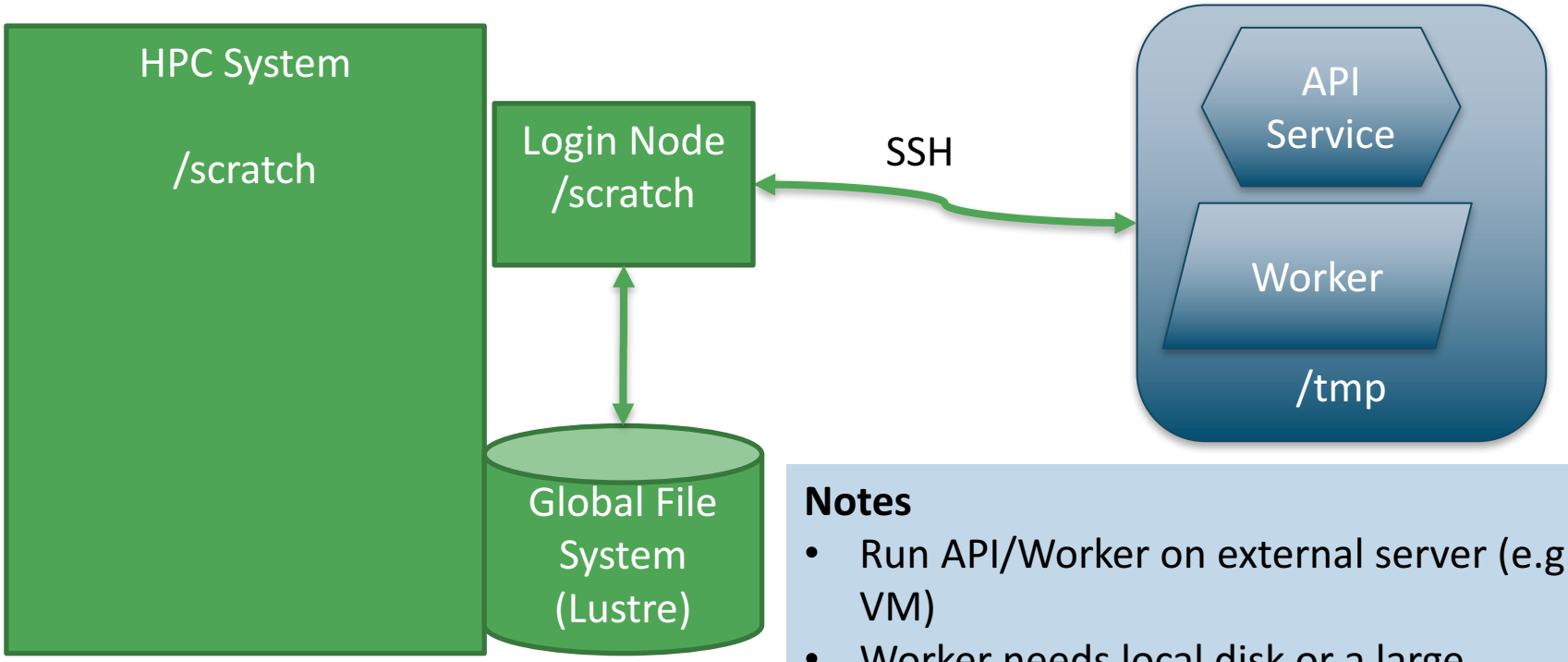
Co-located/Local



Notes

- Typically run on a login-class node
- Worker needs local disk or a large ramdisk space
- This is the easiest model and recommended for most cases

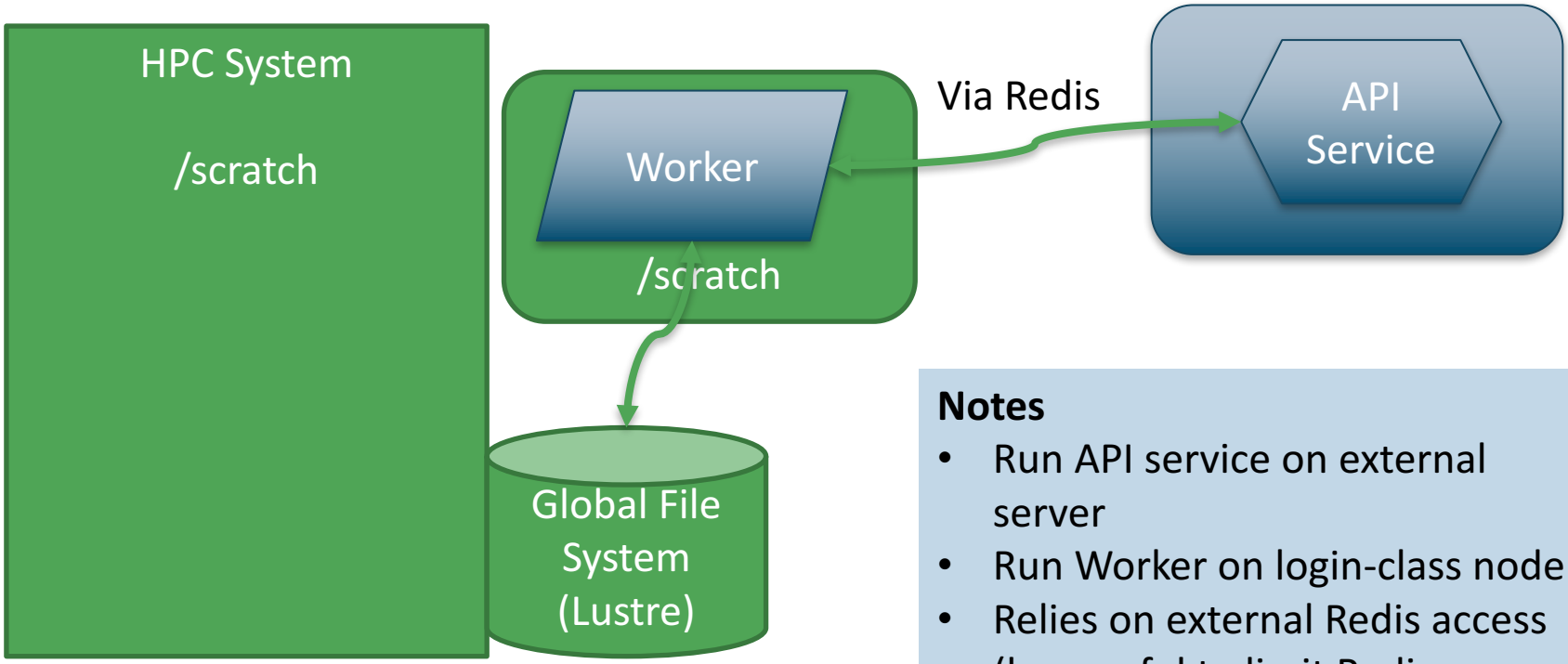
Co-located/Remote



Notes

- Run API/Worker on external server (e.g. VM)
- Worker needs local disk or a large ramdisk space
- Worker needs ssh keys configured to copy image to a node that has the file system mounted.

Split/Local



Notes

- Run API service on external server
- Run Worker on login-class node
- Relies on external Redis access (be careful to limit Redis access)
- Most useful for multi-system deployments

Registries

- Adding registries is done in the image manager
- This can be used to add both private and public registries
- Configured in the Locations seciton of the imagemanager.json config file

Registries

```
…
"Locations": {
        "index.docker.io": {
            "remotetype": "dockerv2",
            "authentication": "http”
        },
"local": {
            "remotetype": "dockerv2",
            "url": "https://localhost:5000",
            "authentication": "http”,
            “sslcacert”: “local.crt”
        }
    },
…
```