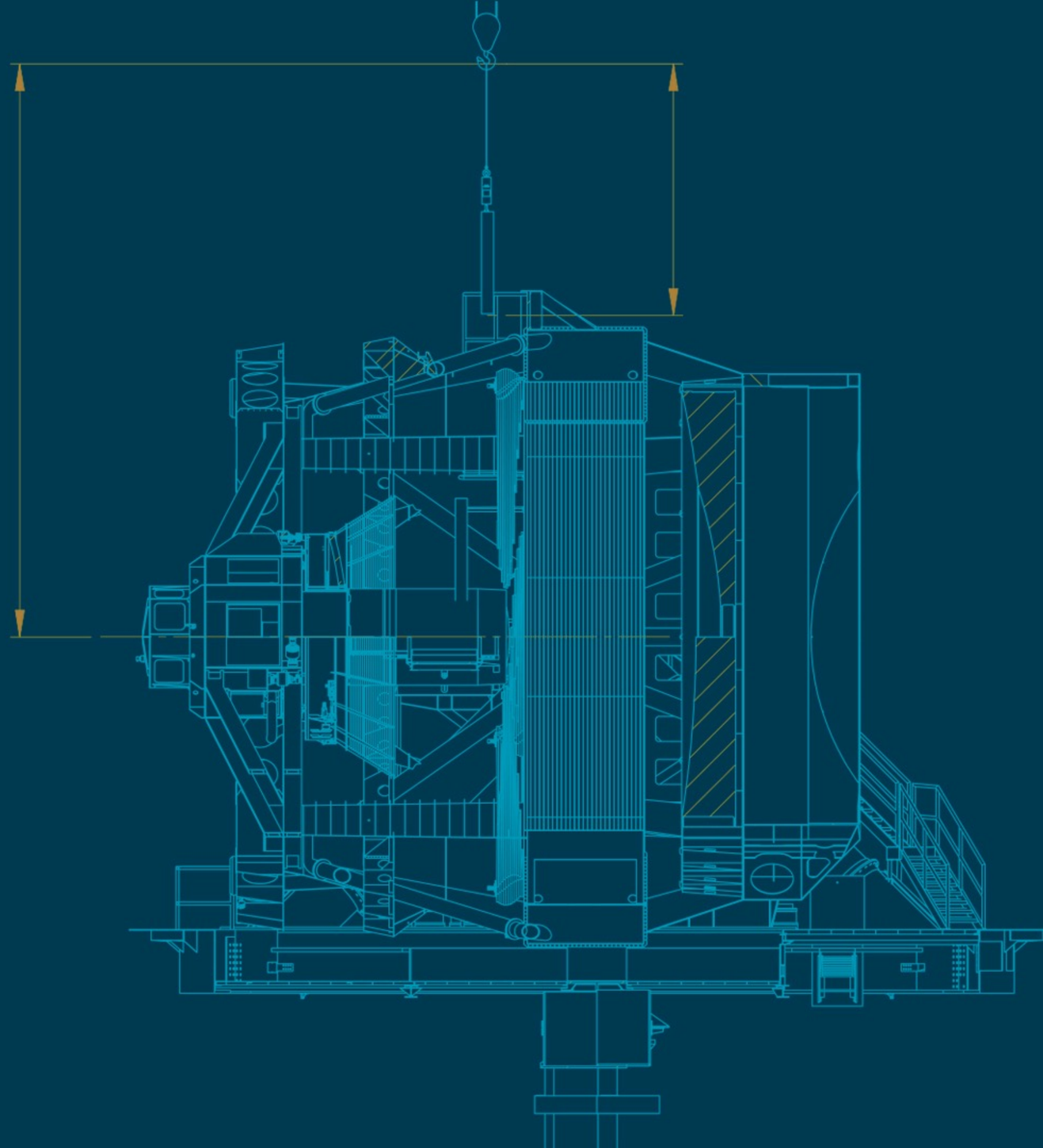
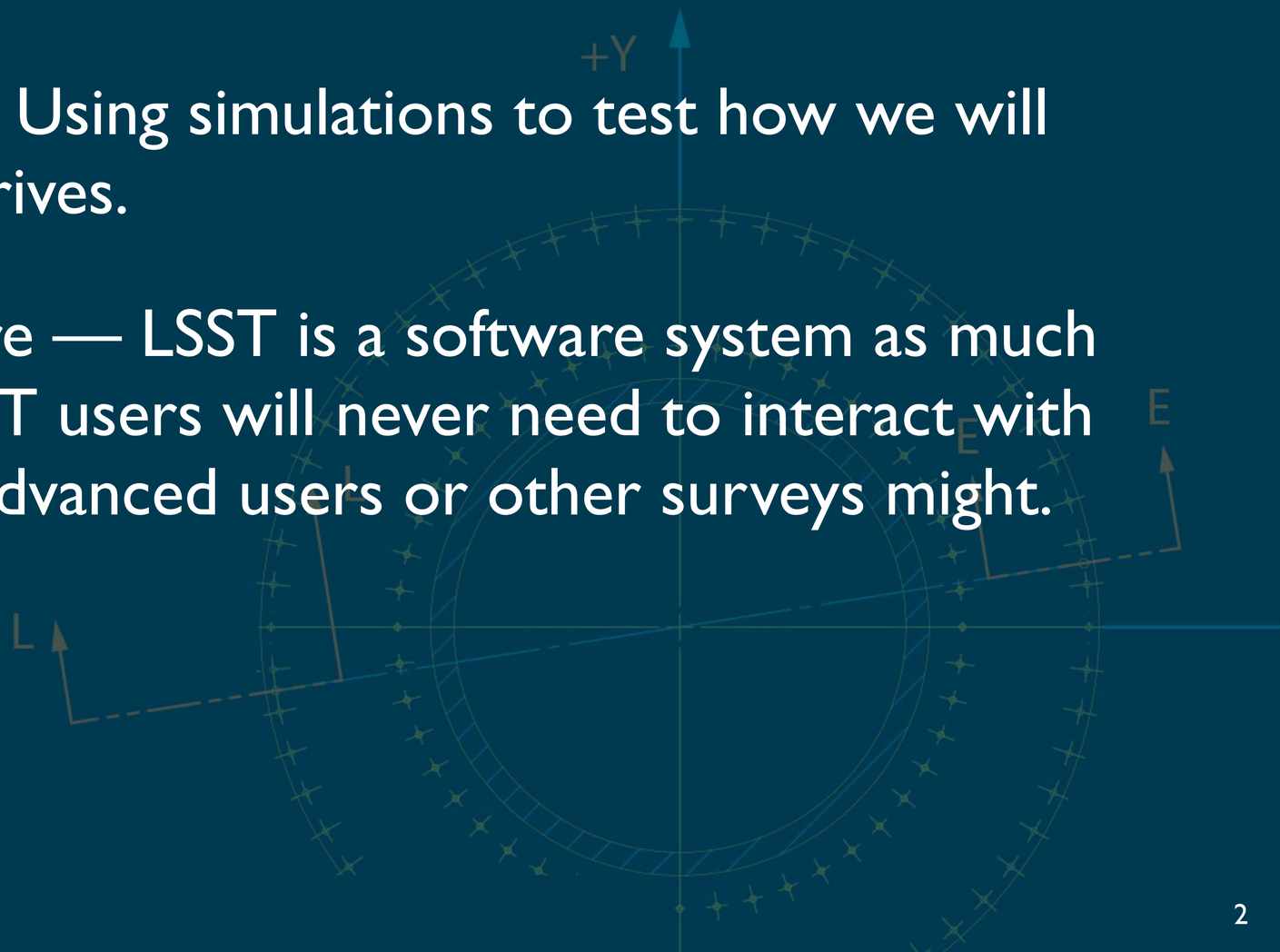
HANDS-ON WITH LSST

Colin Slater
LSST DM Science Group
University of Washington



- Hands on with the data — Using simulations to test how we will handle the data when it arrives.
- Hands on with the software — LSST is a software system as much as it is hardware. Most LSST users will never need to interact with the pipelines directly, but advanced users or other surveys might.



Step 0: Install

```
%  export PATH="/path/to/conda/bin:$PATH" # If needed

%  conda config --add channels http://conda.lsst.codes/stack/0.13.0
%  conda create --name lsst python=2
%  source activate lsst
%  conda install lsst-distrib
%  conda install lsst-obs-decam

%  pip install sncosmo iminuit
%  git clone https://github.com/lsst-dmsst/demo-lsst-uk-2017
```

Hands On the Data

- We've created a set of light curves that simulate the data that LSST will produce.
- These mock realizations are a good way to go from abstract thinking about LSST to concrete questions about doing science.
- We have three types of objects: RR Lyrae, Supernovae, and AGN.

Universe Model: What objects are in the sky, and where?

OpSim: When and Where does the telescope point?

CatSim: What do those objects look like?

Mock Realizations of LSST Data

- Goals of this session:
  - Develop a feeling for the “best case scenario”. 10 years of data with no noise.
  - Gradually increase the realism and the challenge. Add the observational noise. Try using only the first year of data. How faint can you go? What do your output uncertainties look like?

- Topics to think about:
  - Follow up: say you only saw the first few data points; how might you decide whether to follow up an object with spectroscopy?
  - Classification: How well can you distinguish these type of objects?
  - What other tools do you need? What is the next most important data to have?

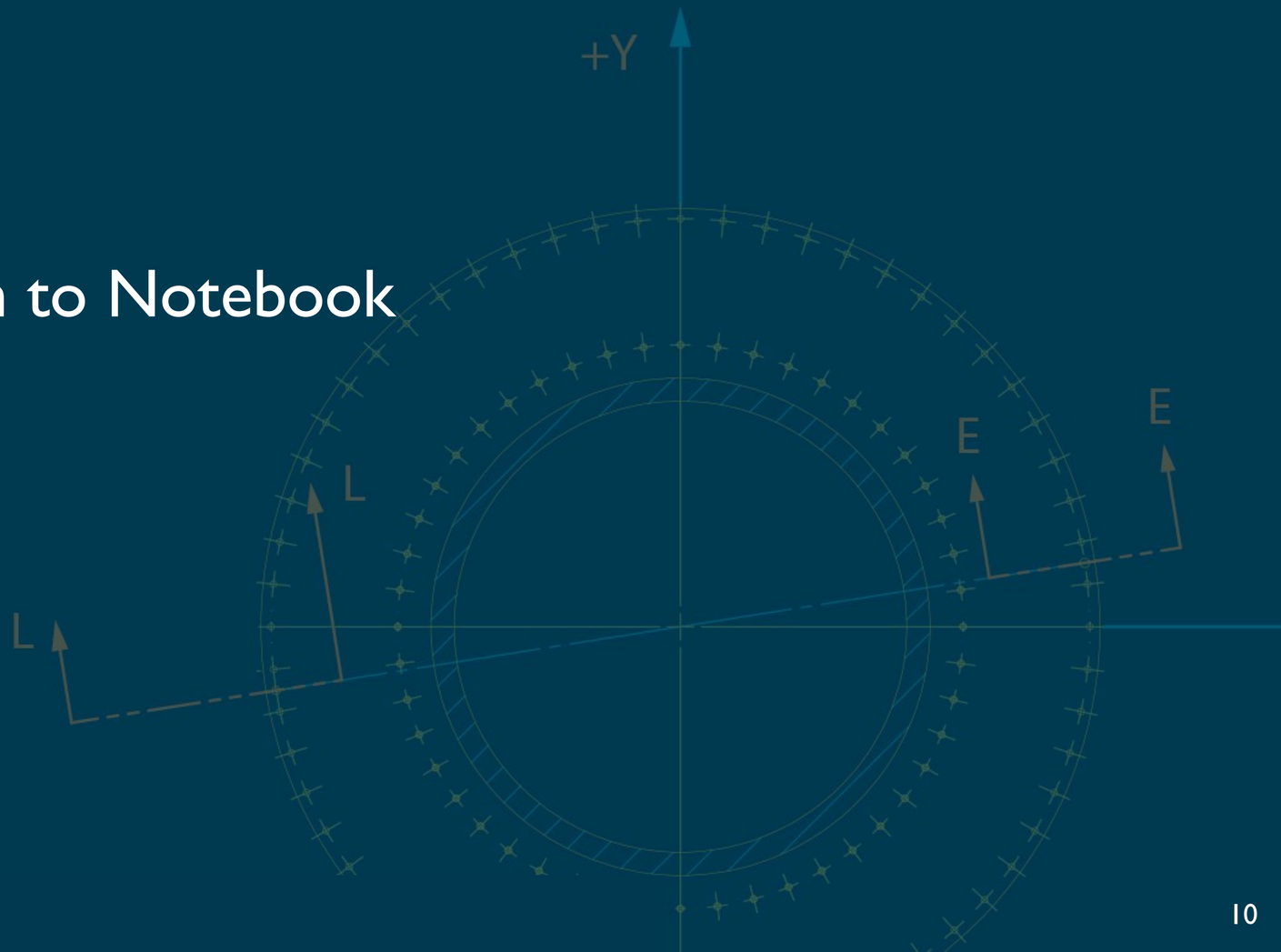
Step 1: Setup & Switch to Notebook

```
%  source activate lsst

%  cd demo-lsst-uk-2017/Lightcurves
%  jupyter notebook
```

csh users: source "$CONDA_PREFIX/bin/eups-setups.csh"

Switch to Notebook



Hands On with the Software

- The majority of LSST users will never need to process raw data.
- But users of other instruments could run the LSST software, or people interested in special processing, or new algorithms.
- We're going to step through a basic part of the processing, from "raw" data to a calibrated image and catalog of measured sources. This step is called "processCcd.py"

Some Definitions:

- "Tasks" — The basic unit of software that does some processing. processCcd.py is a task, and it also calls many "sub-tasks" that perform individual parts of the processing.
- "The Butler" — Software that manages a basic database of the images available on disk, and the output data products.
- Repository — The Butler manages a "repo" of data.
- Re-run — A set of outputs that were processed together.

Outline of Steps:

1. "setup" the obs_decam package, which loads the rest of the LSST stack

2. "Ingest" the input images into the Butler's database. (Image, weight map, and mask)

3. Configure processCcd.py. We need to set the type of input data and what processing steps we want to do.

4. Run processCcd.py on one CCD. This will create an output "calexp" FITS image and a catalog of sources.

Step 1: Setup

```
%  source activate lsst
%  source eups-setups.sh
%  setup obs_decam

%  cd demo-lsst-uk-2017/ProcessCcd
%  wget http://lsst-web.ncsa.illinois.edu/~ctslater/
decam_visit.tar.gz
%  tar xzf decam_visit.tar.gz       # Creates decam_data/
%  mkdir repo
%  echo lsst.obs.decam.DecamMapper > repo/_mapper
```

csh users: source "$CONDA_PREFIX/bin/eups-setups.csh"

Step 2: Ingest

```
% ingestImagesDecam.py repo --filetype instcal \
  decam_data/instcal/*.fits.fz
```

Step 3: Configuration

```
%  cat my_config.py
> from lsst.obs.decam.decamNullIsr import DecamNullIsrTask
> config.isr.retarget(DecamNullIsrTask)
>
> config.calibrate.doAstrometry=False
> config.calibrate.doPhotoCal=False
```

Step 4: Run

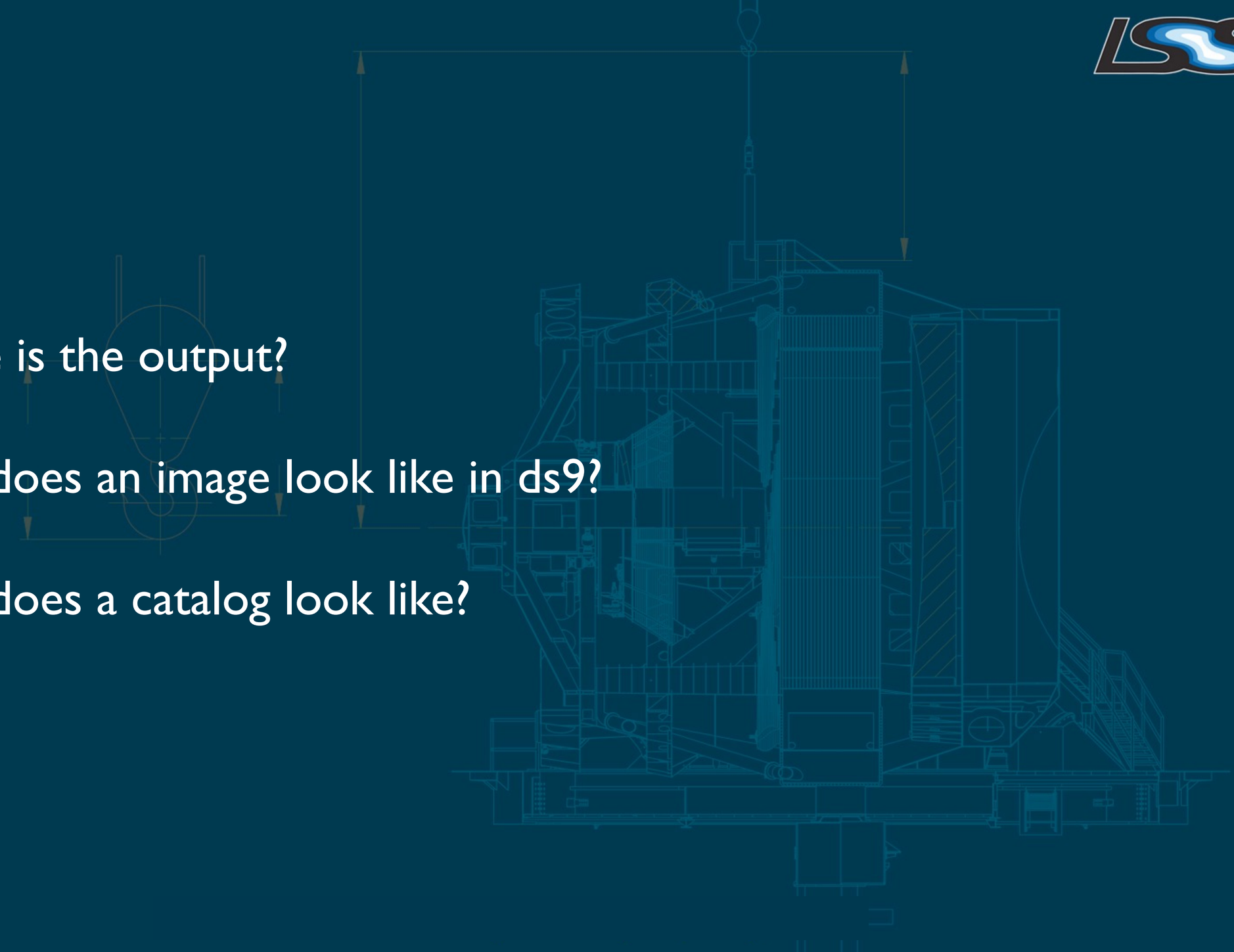
```
% processCcd.py repo --id visit=303527 ccdnum=10 \
    -C my_config.py --rerun my_reduction
```

Step 4: Run

```
%  processCcd.py repo --id visit=303527 ccdnum=10 \
    -C my_config.py --rerun my_reduction
```

```
> processCcd.calibrate.deblend INFO: Deblended: of 1144 sources, 116 were deblended
creating 284 children, total 1428 sources
> processCcd.calibrate.measurement INFO: Measuring 1428 sources (1144 parents,
284 children)
> processCcd.calibrate.applyApCorr INFO: Applying aperture corrections to 2 flux
fields
> processCcd.calibrate.applyApCorr INFO: Use naive flux sigma computation
> processCcd.calibrate INFO: Copying flags from icSourceCat to sourceCat for 220
sources
```

Success!



- Where is the output?
- What does an image look like in ds9?
- What does a catalog look like?