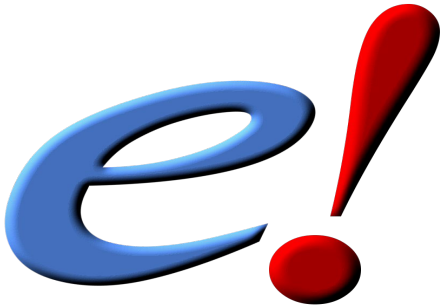
The eHive system for pipeline management and execution

Part 1: System overview

Instructors:
Leo Gordon and Brandon Walts

EMBL-EBI

wellcome trust sanger institute



http://training.ensembl.org/events/2017/2017-03-23-ehiveRoslin

Course overview

Day 1	Introduction to eHive
	Initializing and running eHive pipelines
	Pipeline configuration: modifying existing pipelines, and writing your own pipelines (part 1)
Day 2	Pipeline configuration (part 2)
	Writing your own runnable modules

Audience background

How many of you have experience:

- Running a pipeline or workflow
  - “By hand”
  - With a shell script
  - With a pipeline management tool
    (Galaxy, Taverna, CWL, JobTree...)
- Submitting jobs to a compute cluster
  (using LSF, SGE)
  - Using job control features of the scheduler
    (job arrays, wait-for rules)
- Writing SQL queries
- Coding in Perl

eHive: an overview and brief history

At Ensembl, we run lots of workflows.

These pipelines have been getting more complex and running longer as the amount of data expands

We also have workflows that may not be complicated, but need to run in an automated way

In 2004, eHive was created to provide a standard, robust way to create and run analysis pipelines

Design goals:

- Scalable
- Adaptable
- Reproducible
- Traceable

eHive provides:

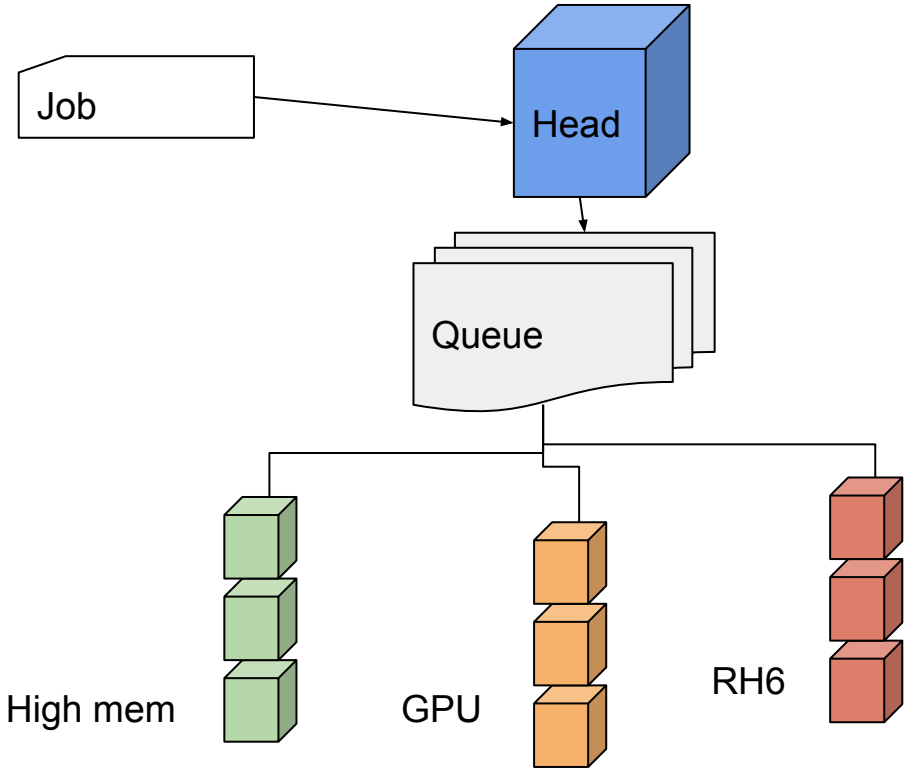
A framework to define workflows

- Sequencing operations
- Defining dependencies
- Flowing data
- Assigning resources

A system for executing workflows

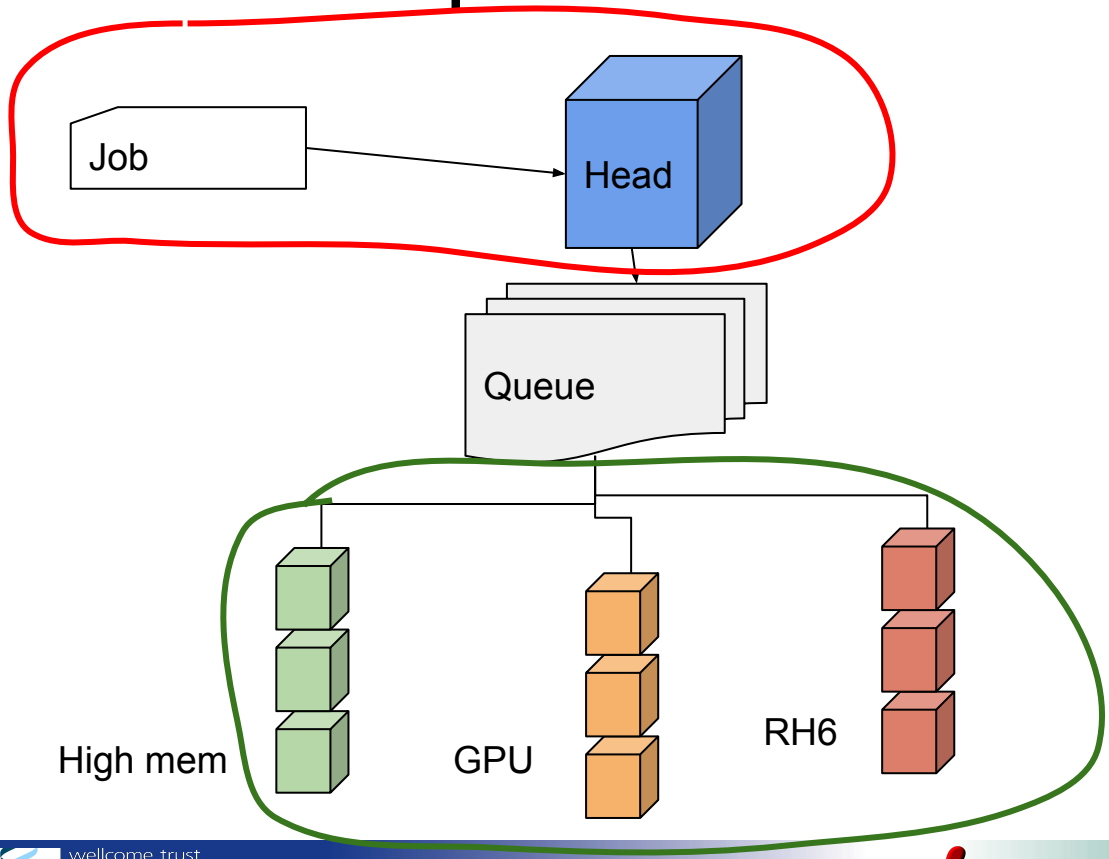
- Job management
- Logging
- Checkpointing

Compute farms



- User at the head node submits job into a queue
- Scheduler assigns job onto a compute node based on
  - User priority
  - Resource requirements
  - Load
- More accurate load estimate = better chance to be scheduled
- Overhead for scheduling a job
- Some basic ability to handle relationships between jobs
  - Arrays
  - Wait-for

eHive's place in the farm



- eHive does not replace the scheduler (LSF, SGE, SLURM, etc.).
- It works one layer above the scheduler
- It is an automated, intelligent job submission engine
- It is an automated, intelligent job execution engine

eHive fundamentals

eHive's model of distributed computation

- Independent agents perform computation
- Coordination via a central list of jobs
- Agents are responsible for claiming jobs, performing work, and updating the central list

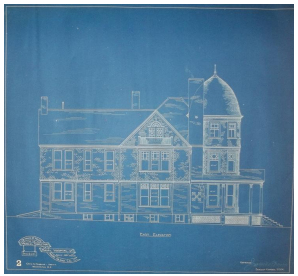
eHive fundamentals: walkthrough

Let’s start with an analogy, building a house

image: David Wright, cc license

eHive fundamentals: walkthrough

What do we need to get our task done:



Blueprints: an overall plan. This plan is fixed and changes rarely if at all over the course of the project.

Job list: the current state of what has been done and what needs to be done next. This is being updated continuously as the project progresses.

Supervisor

Workers

Resources

Procedures: exactly how to perform a job

eHive fundamentals: walkthrough

The difference between the blueprints, the job list, and procedures

Blueprints are:

- Fixed throughout the whole project (in general)
- Describe “classes” of jobs
- Describe dependencies between job classes
- Describe resources required for jobs

Job lists are:

- Dynamic, jobs are being added throughout the project
- Describe “instances” of jobs
- Describe dependencies between individual jobs

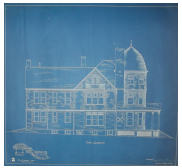
eHive fundamentals: walkthrough

The difference between the blueprints, the job list, and procedures

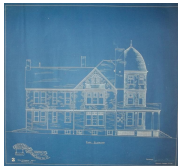
Procedures are:

- Fixed throughout the project
- Describe step by step how to do a particular job
- Are parameterized, so they can be customized by plugging in different parameters

eHive fundamentals: walkthrough



eHive fundamentals: walkthrough



eHive fundamentals: walkthrough

In eHive, the supervisor is very “hands off” and the workers take care of managing the jobs themselves.

Let’s follow a worker to see how this happens...

eHive fundamentals: walkthrough

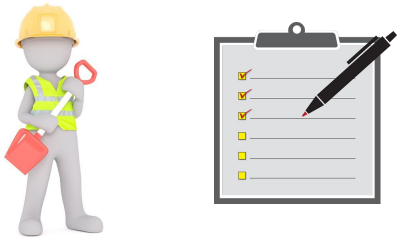
follow a worker...

1. Worker checks the job list and finds a job it can do

2. Worker claims a job, and specializes to perform it

eHive fundamentals: walkthrough

follow a worker...

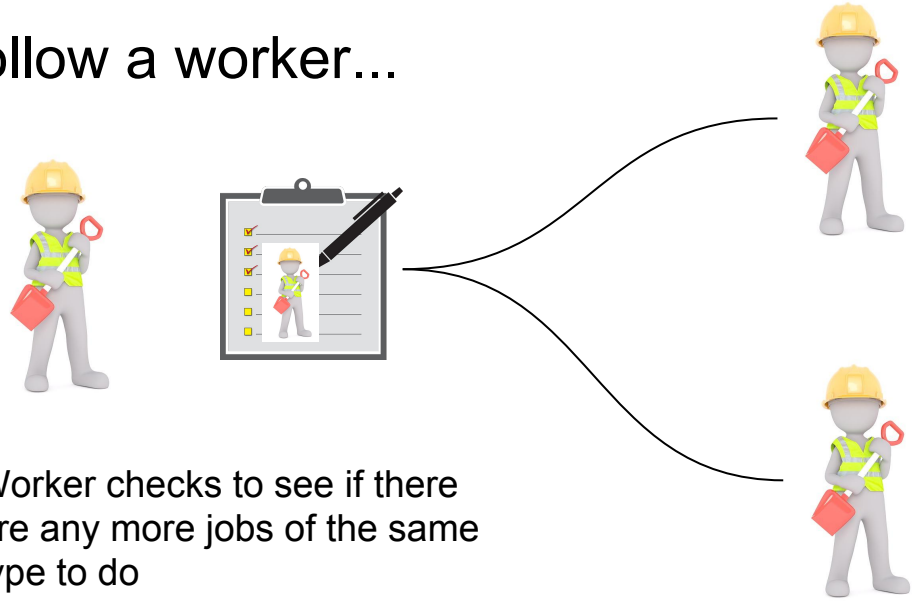


3. Worker finishes job and marks it done on the job list

4. **Worker** checks the blueprints to see if it needs to add any new jobs to the list

eHive fundamentals: walkthrough

follow a worker...



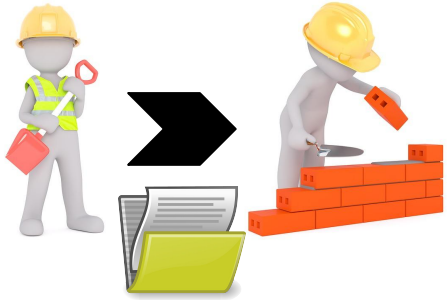
5. Worker checks to see if there are any more jobs of the same type to do

6. If so, worker claims another job and starts to work on it

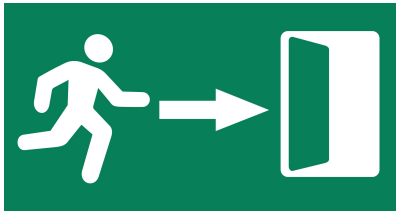
7. If not, worker checks to see if there are other types of jobs it can do

eHive fundamentals: walkthrough

follow a worker...



8. If so, worker claims the new job and respecializes to perform it



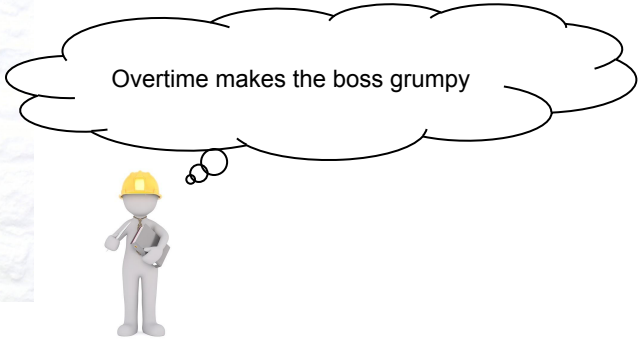
9. If not, worker leaves

eHive fundamentals: walkthrough

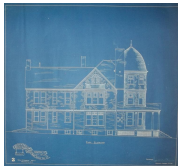
Worker time limit

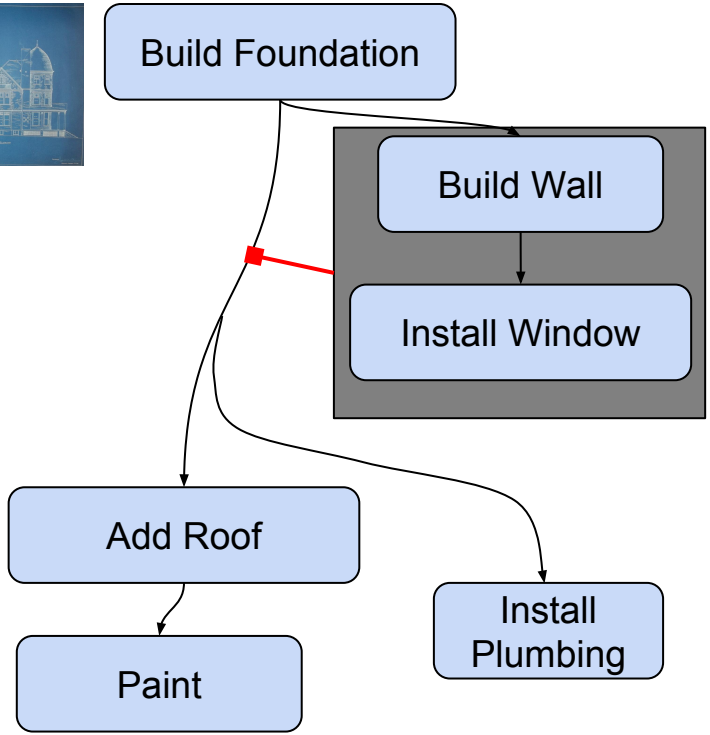
Workers have a time limit:

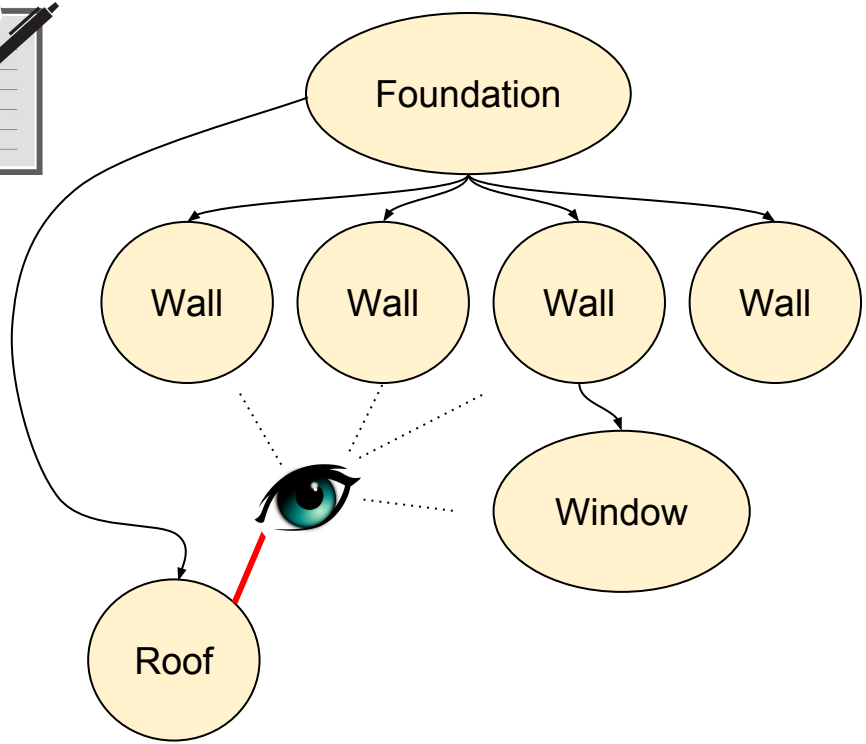
- Default 1 hour
- After reaching limit, workers exit
- However, a worker will never abandon a job in progress



eHive fundamentals: walkthrough



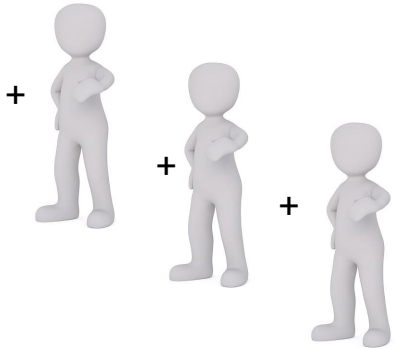




eHive fundamentals: walkthrough

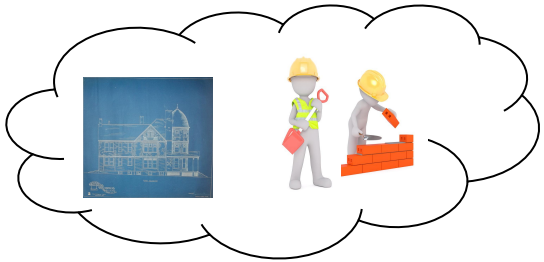
The role of the supervisor

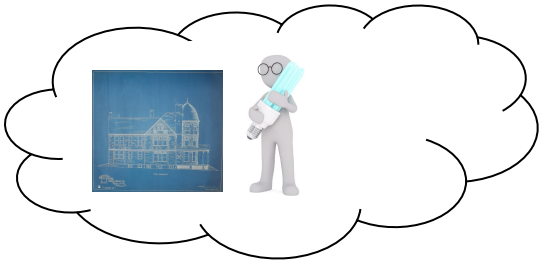
The supervisor looks at the current job list, blueprints, and staff, and hires more workers if needed.



eHive fundamentals: walkthrough

A bit about specialization





- A worker knows about its capabilities.
- When looking for a job to claim, it checks the requirements for that class of job.
- If the worker's capabilities meet the job's requirements, it claims that job.
- A worker doesn't know "I can install lightbulbs." It knows "I can reach high" and it sees "install lightbulbs jobs have a "can reach high" requirement, so it knows it can specialize

eHive fundamentals: walkthrough

There has to be at least one job to get started

So there's a special mechanism to create at least one job when starting the project - usually this is the first class of job, but it can start anywhere

eHive fundamentals: components

How this model is actually implemented in eHive

eHive fundamentals: components

Blueprints

Job list

Hive database

Can be run in any of:

MySQL® PostgreSQL SQLite

eHive fundamentals: components

Blueprints

Job list

Hive database

Procedures

=

```
use base
('Bio::EnsEMBL::Hive::Process')

{
}
```

eHive fundamentals: components

Blueprints

Job list

Hive database

Procedures

```
use base
('Bio::EnsEMBL::Hive::Process')

{
}
```

Worker

=

Running process on a local machine or a farm node

eHive fundamentals: components

Blueprints

Job list

Hive database

Procedures

=

```
use base
('Bio::EnsEMBL::Hive::Process')

{
}
```

Worker

=

Running process on a local machine or a farm node

Supervisor

=

Running process on a local machine

eHive fundamentals: hive database

```
use base
('Bio::EnsEMBL::Hive::PipeConfig::
HiveGeneric_conf')

{
}
```

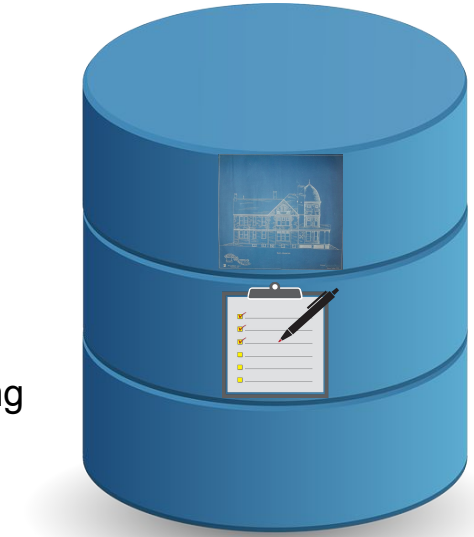
PipeConfig

init_pipeline.pl

Blueprints

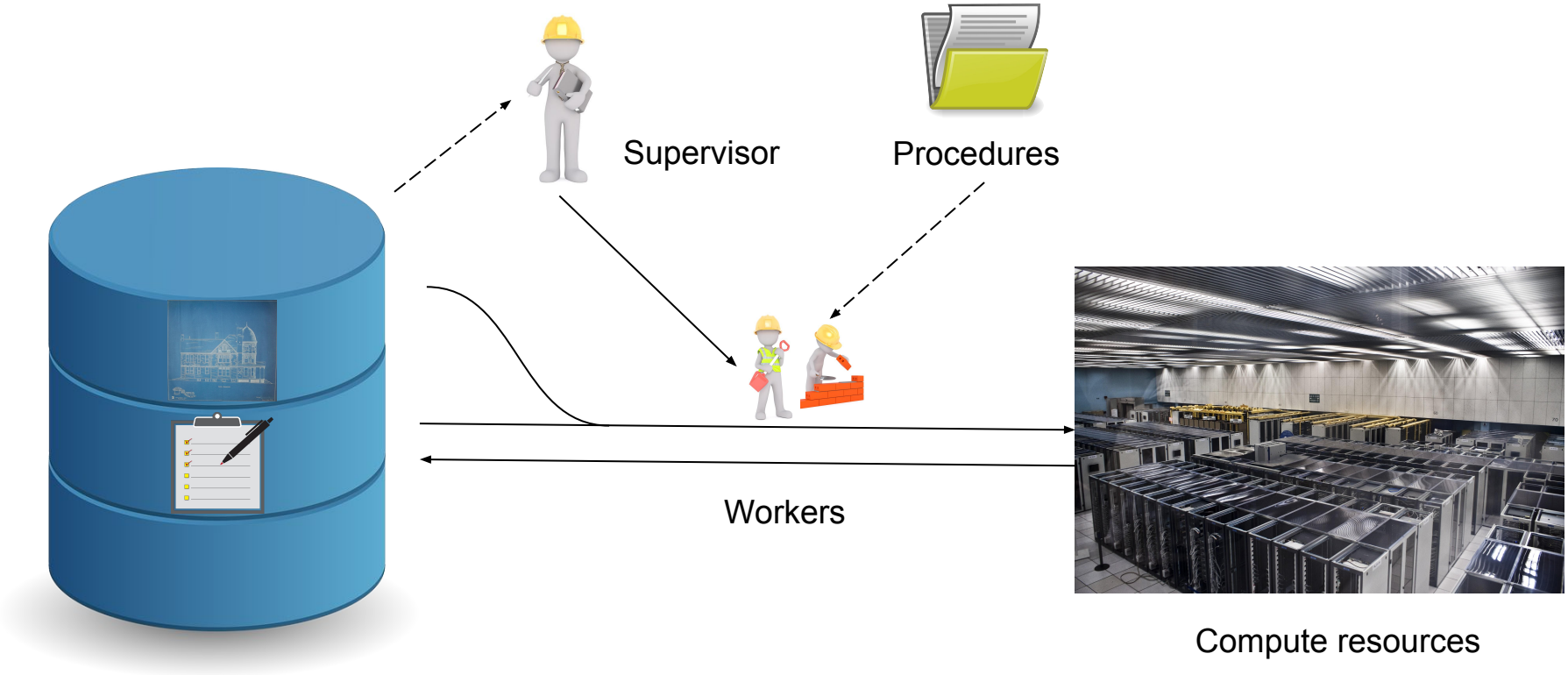
Job list

Other housekeeping and logs



Hive database

eHive fundamentals: running



CC image by F. Hirzinger

eHive fundamentals: terminology

Blueprints stored in a file

PipeConfig

Supervisor

Beekeeper

Worker

Worker

Procedure

Runnable

Compute resources

Meadow (~ a farm head node) and Valley (group of meadows)

Hive database

Hive database (details covered later in the course)

Ensembl Acknowledgements

The Entire Ensembl Team

Bronwen L. Aken[1], Premanand Achuthan[1], Wasiu Akanni[1], M. Ridwan Amode[1], Friederike Bernsdorff[1], Jyothish Bhai[1], Konstantinos Billis[1], Denise Carvalho-Silva[1], Carla Cummins[1], Peter Clapham[2], Laurent Gil[1], Carlos García Girón[1], Leo Gordon[1], Thibaut Hourlier[1], Sarah E. Hunt[1], Sophie H. Janacek[1], Thomas Juettemann[1], Stephen Keenan[1], Matthew R. Laird[1], Ilias Lavidas[1], Thomas Maurel[1], William McLaren[1], Benjamin Moore[1], Daniel N. Murphy[1], Rishi Nag[1], Victoria Newman[1], Michael Nuhn[1], Chuang Kee Ong[1], Anne Parker[1], Mateus Patricio[1], Harpreet Singh Riat[1], Daniel Sheppard[1], Helen Sparrow[1], Kieron Taylor[1], Anja Thormann[1], Alessandro Vullo[1], Brandon Walts[1], Steven P. Wilder[1], Amonida Zadissa[1], Myrto Kostadima[1], Fergal J. Martin[1], Matthieu Muffato[1], Emily Perry[1], Magali Ruffier[1], Daniel M. Staines[1], Stephen J. Trevanion[1], Fiona Cunningham[1], Andrew Yates[1], Daniel R. Zerbino[1] and Paul Flicek[1,2,*]

[1]European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Genome Campus, Hinxton, Cambridge CB10 1SD, UK and [2]Wellcome Trust Sanger Institute, Wellcome Genome Campus, Hinxton, Cambridge, CB10 1SA, UK

Funding

wellcome

EMBL

National Human Genome Research Institute

Open Targets

BBSRC bioscience for the future

TRANSFORMING GENETIC MEDICINE INITIATIVE

MedBioinformatics

Co-funded by the European Union