The Elizabeth H.
and James S. McDonnell III
McDonnell Genome Institute
at Washington University

PTero

History, Architecture and Development of a scalable workflow system.

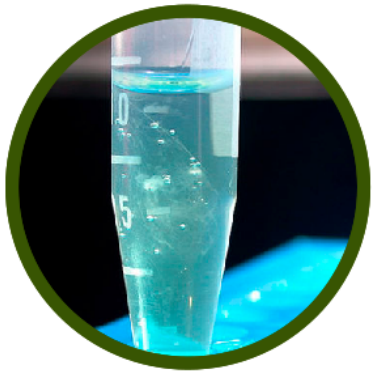
What you're in for...

- History and Background (~15 min)
  - Genome Modeling System -> Workflow -> Flow -> PTero
- PTero Architecture (~15 min)
  - Services
  - Service Interactions
  - Realistic Example
- Development and Testing (~5 min)
- Deployment (~2 min)
- Live Demo? (~3 min)

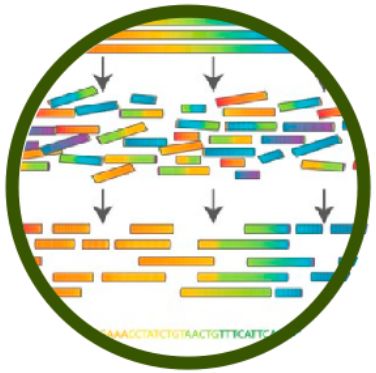
History and Background

Genetic Sequencing and Analysis at MGI

Sample Preparation

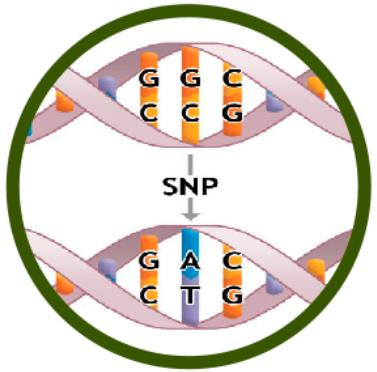


Alignment



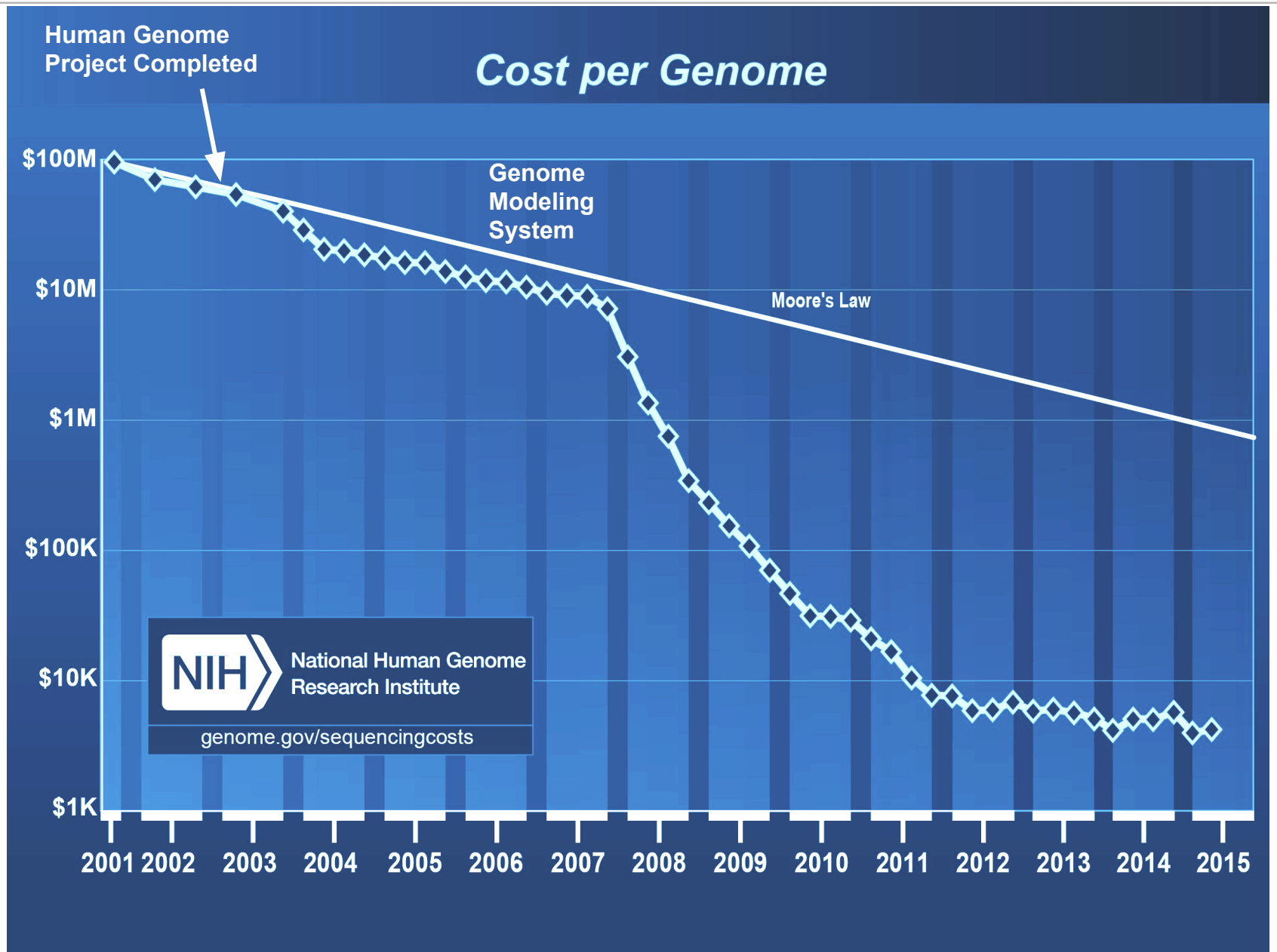
Annotation

Sequencing



Variant Detection

History and Background



History and Background

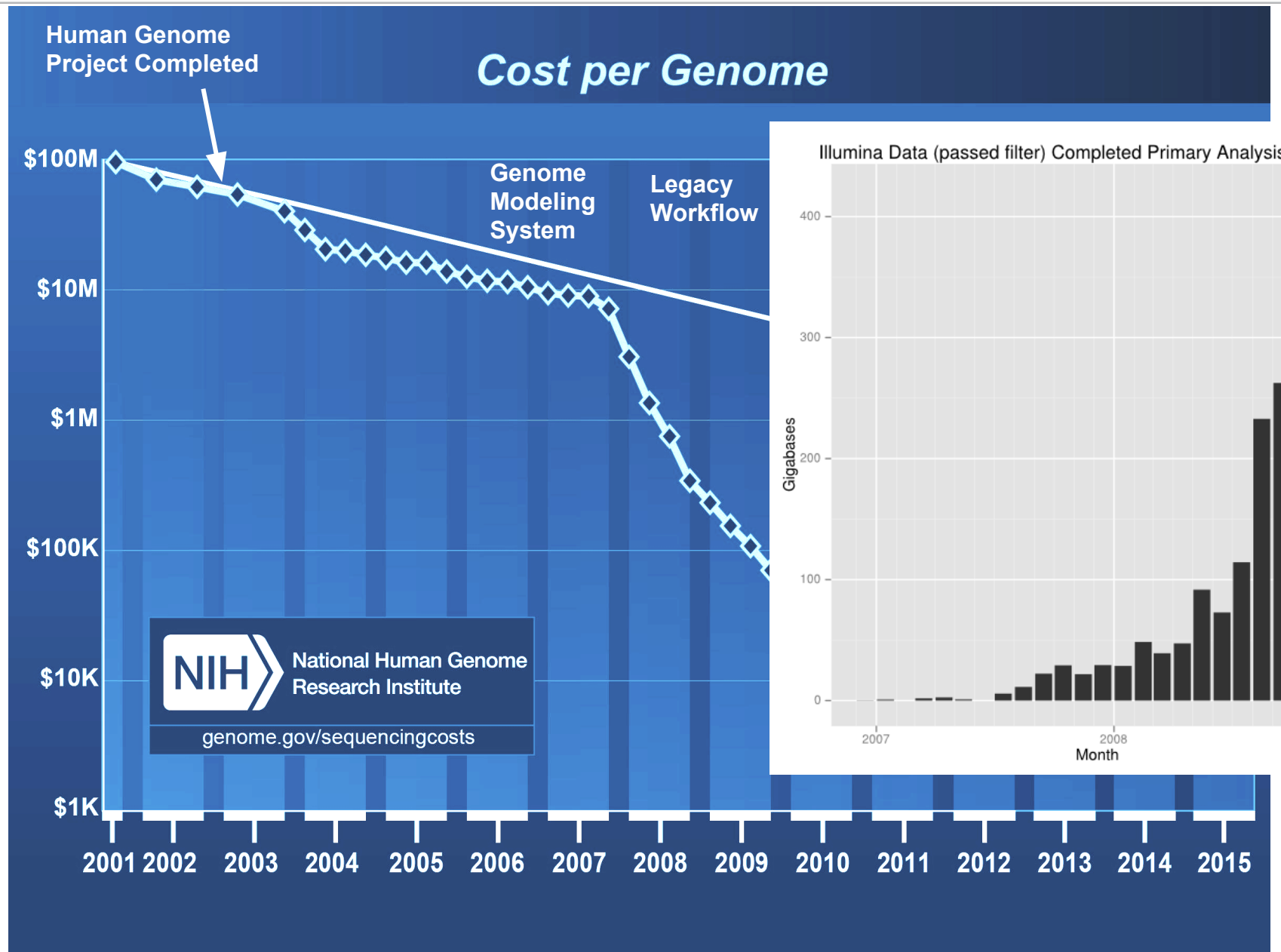
File Based Programs

- Long Running
- Noninteractive
- 3rd Party

Genome Modeling System

- Wrap Programs in Perl
- Expose Command Line Interface
- Create Automated Pipelines

History and Background

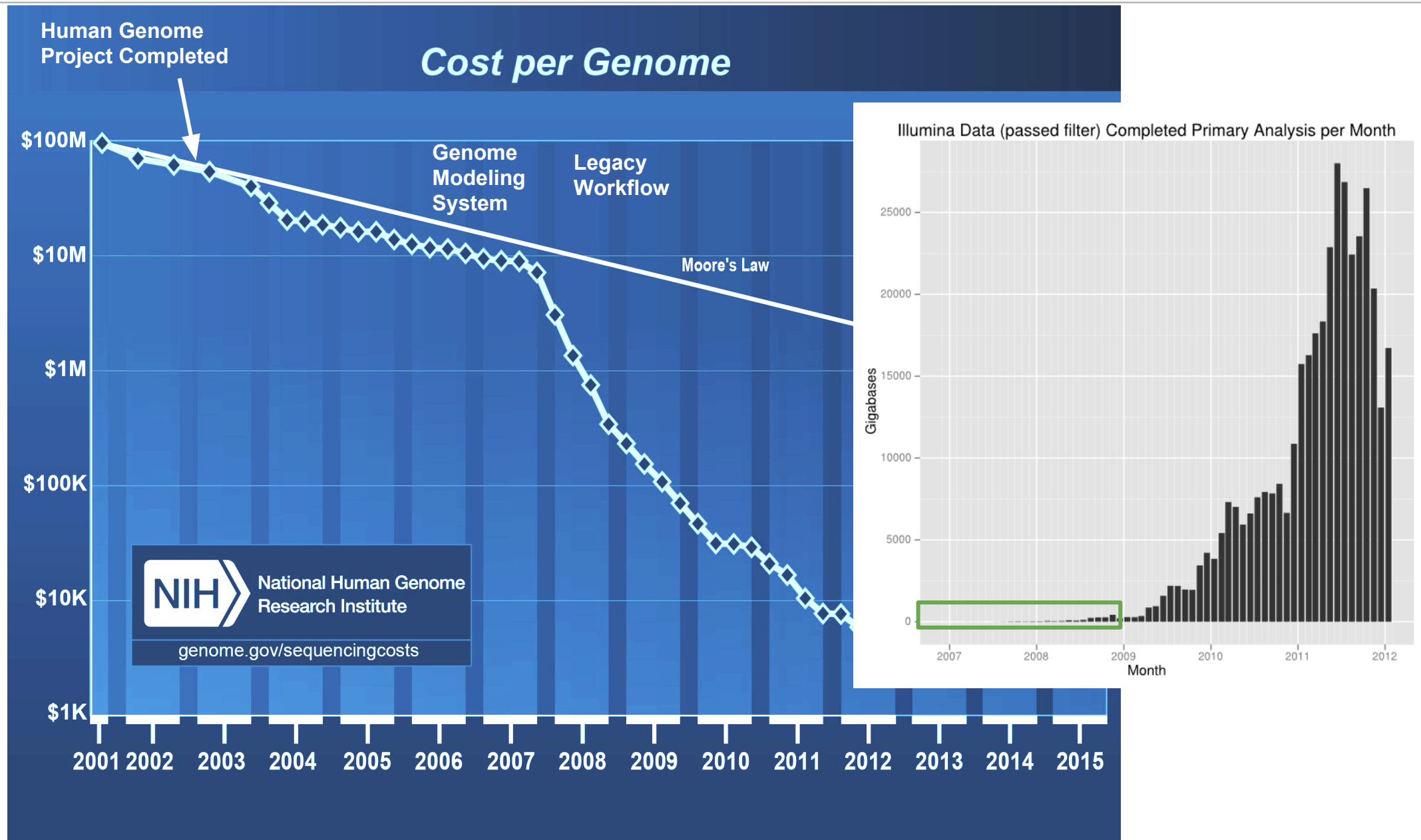


History and Background

Legacy Workflow

- Declarative (xml based workflow description)
- Integrates with datacenter management platform (IBM's LSF)
- Introduces concept of parallel-by
- Has centralized workflow tracking

History and Background



History and Background

Legacy Workflow

Features

- Declarative (xml based workflow description)
- Integrates with datacenter management platform (IBM's LSF)
- Introduces concept of parallel-by
- Has centralized workflow tracking

Limitations

- Cannot scale up
- Highly coupled to Perl, UR and GMS
- Represents workflows as DAGs

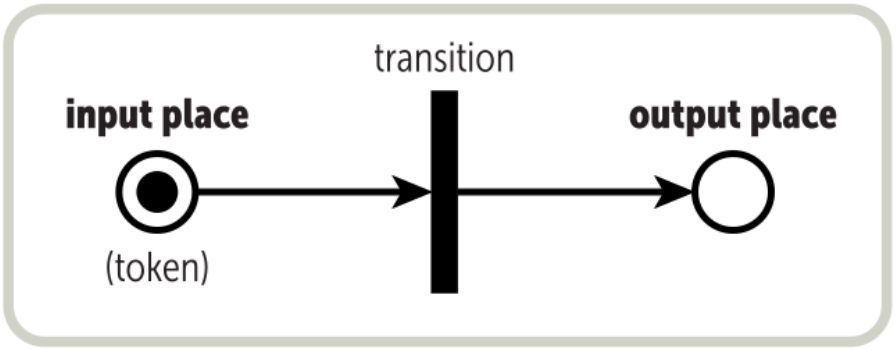
History and Background

	Dependency	Concurrency	Choice
DAG	Yes	Yes	No
FSM	Yes	No	Yes
P/T	Yes	Yes	Yes

Terms

DAG - Directed Acyclic Graph
FSM - Finite State Machine
P/T - Place/Transition or Petri Net

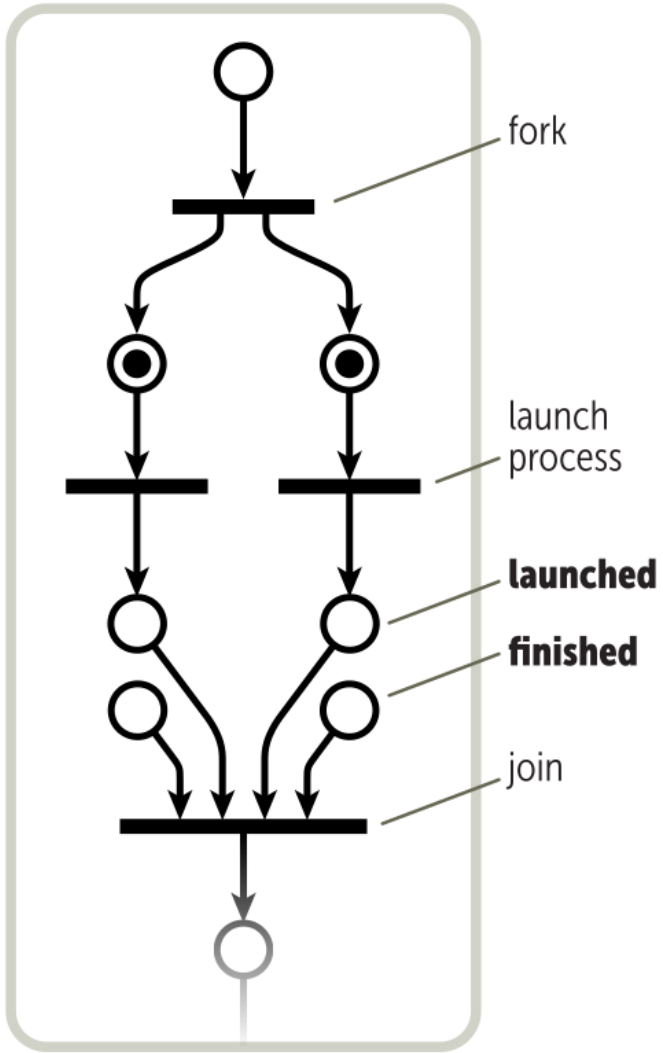
History and Background



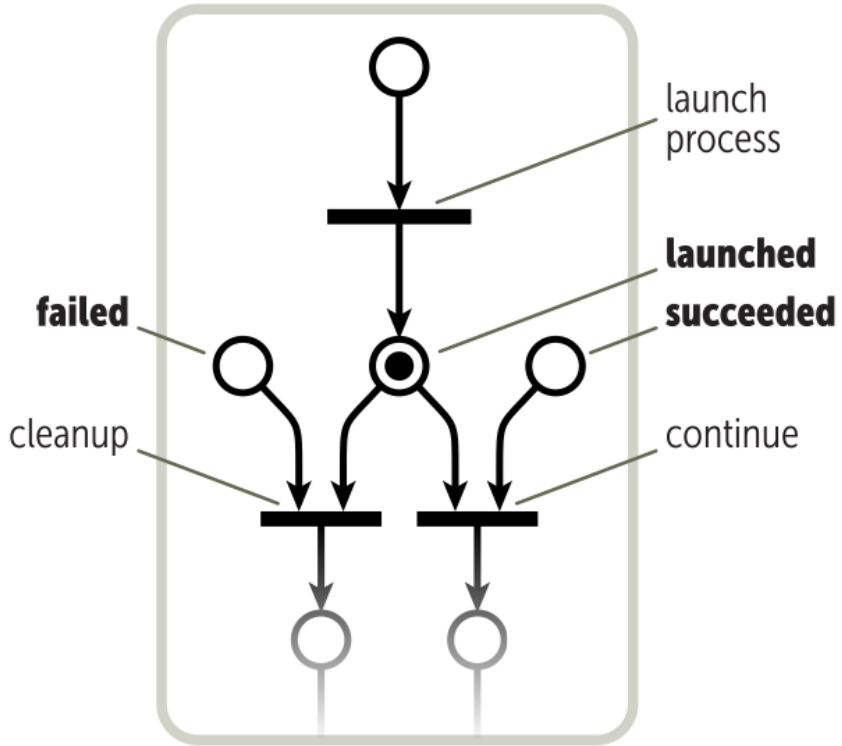
Petri Net Basics

- Places represent **states.**
- Transitions perform **actions.**
- Transitions fire iff all input places have a token.
- After a transition fires, one token is placed in each output place.

History and Background



Concurrency



Choice

History and Background

Flow

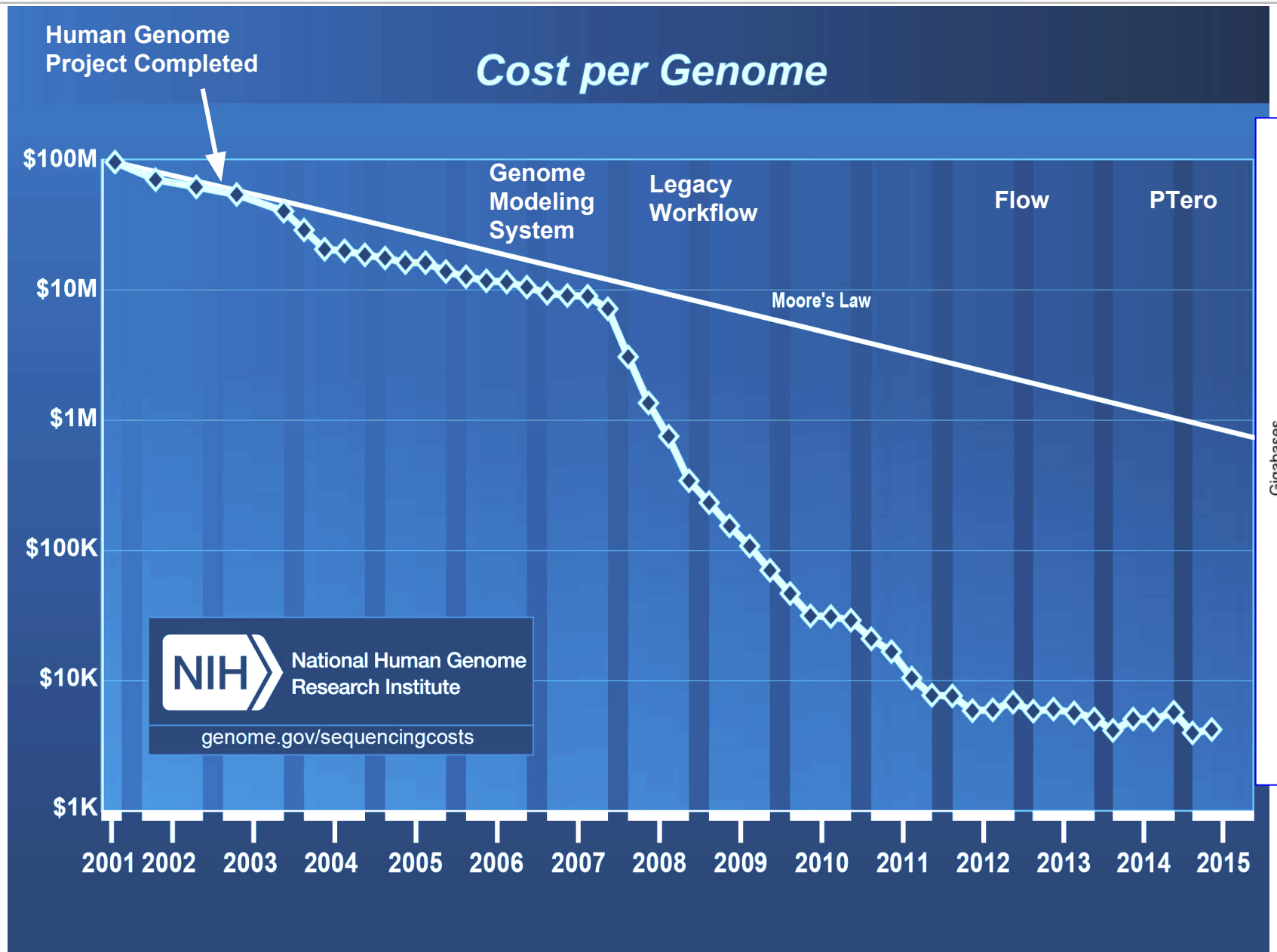
Features

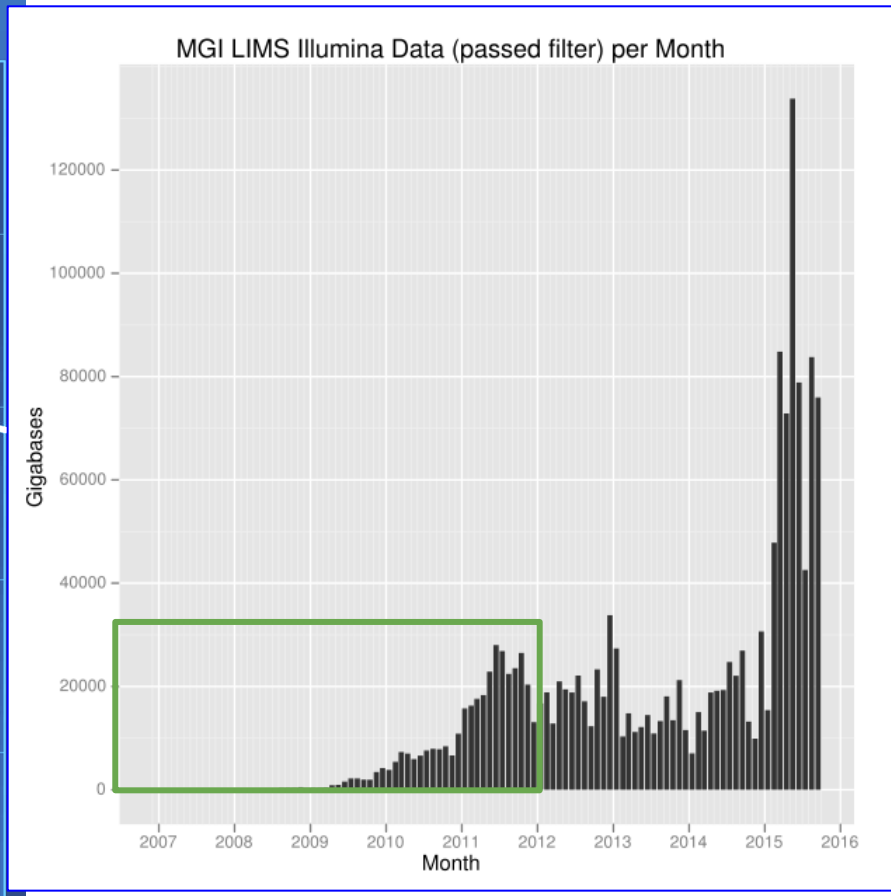
- Scales up nicely
  - Number of concurrent workflows
  - Number of tasks in a workflow
- Service Oriented
- High Reliability

Limitations

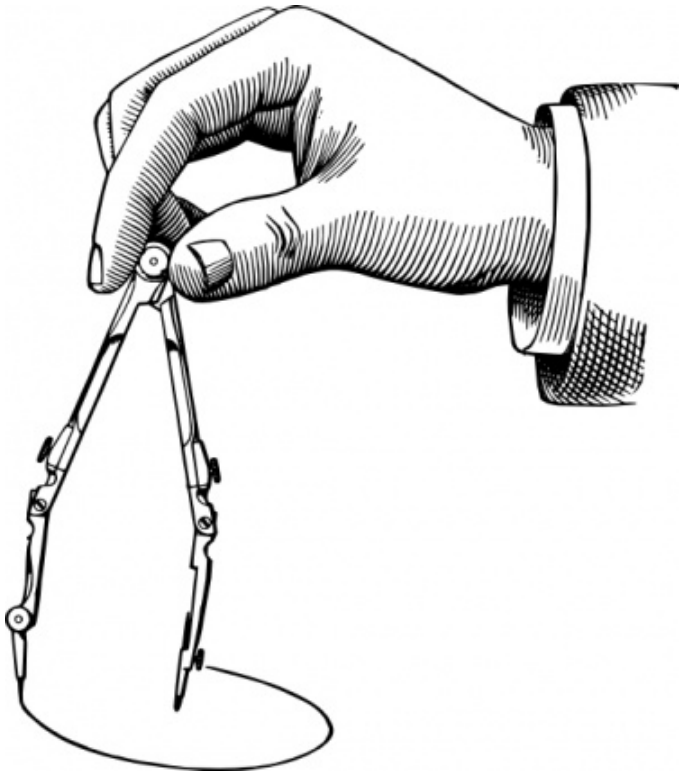
- Heterogeneous and complicated APIs
- Legacy workflow ties
  - Workflow status stored in old DB
  - Workflows represented with same xml

History and Background





History and Background



PTero Design Goals

- Same ability to scale as Flow
- Service oriented with simple RESTful APIs
- JSON based workflow description
- Pluggable job execution services
- Outside adoption

PTero Architecture

Petri Service

- Execute a Petri net

Workflow Service

- Execute a workflow
- Instigate interactions with Petri service and Job services
- Manage task inputs and outputs

Job Services

ShellCommand:

- Execute a shell command given a user, working directory and environment

LSF:

- Submit a job as a given user specifying resources and other LSF options

PTero Architecture

Terminology:

Task - Something to complete

Input/Output - A JSON data structure

Workflow - A DAG of Tasks + inputs

Method - A way to finish a Task

come in two varieties:

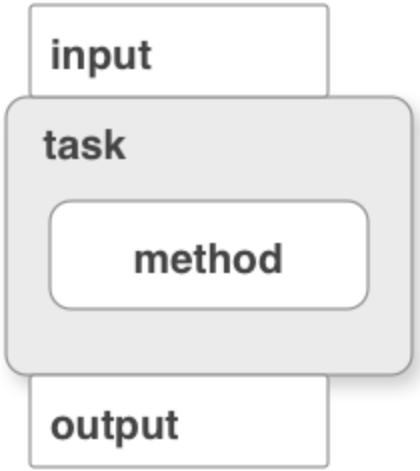
Job - type of method that involves a job service

DAG - type of method that is a DAG of Tasks

Execution - A record of running a Method or Task

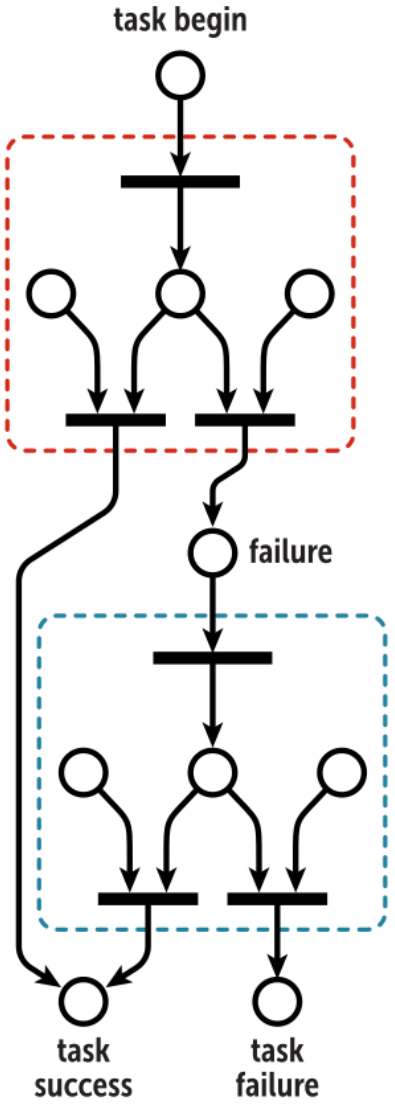
PTero Architecture

Task Notation

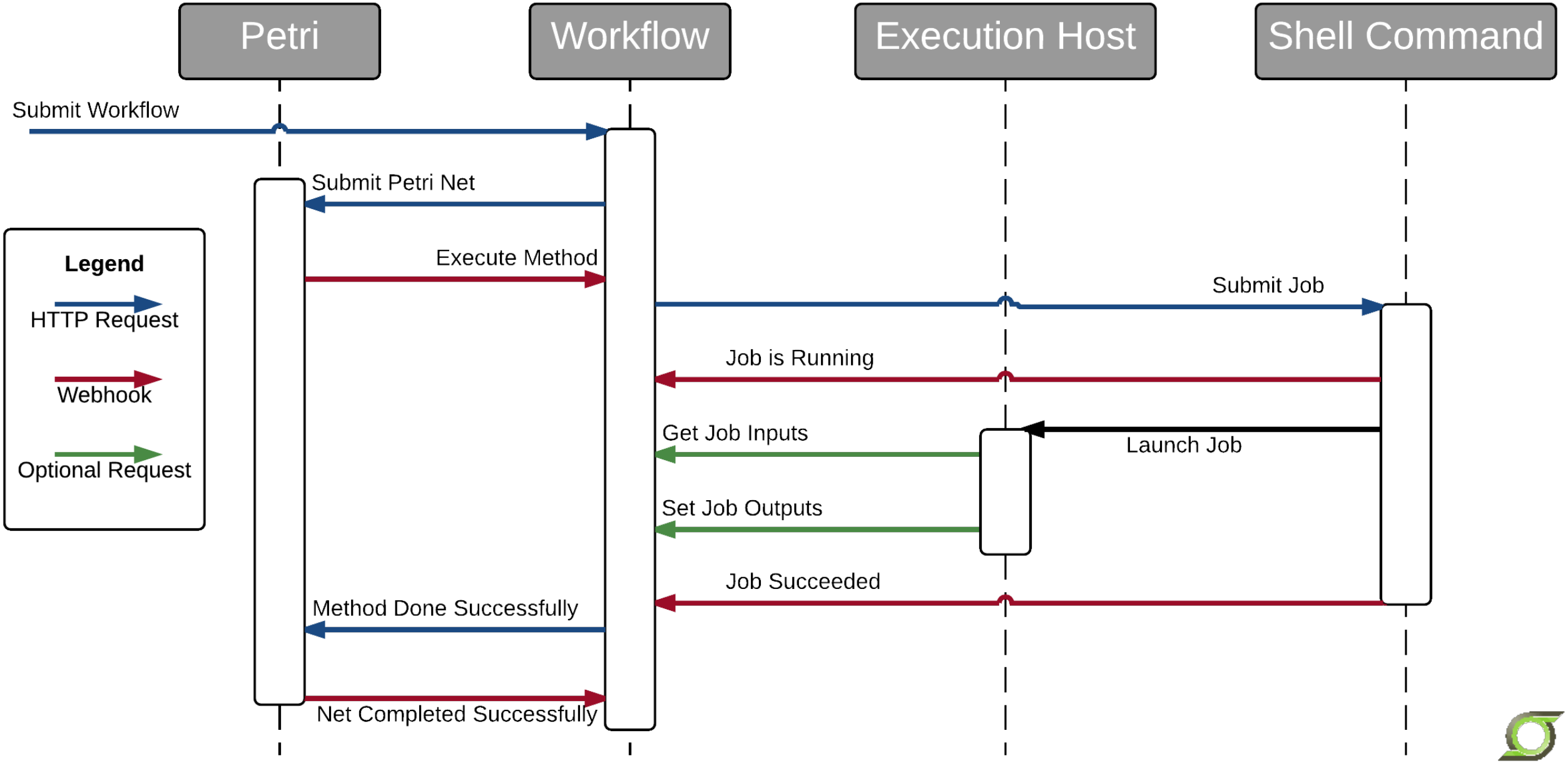


PTero Architecture

Translating a Task into a Petri Net

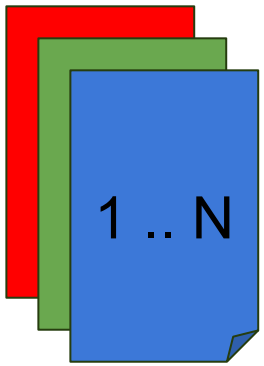


PTero Architecture

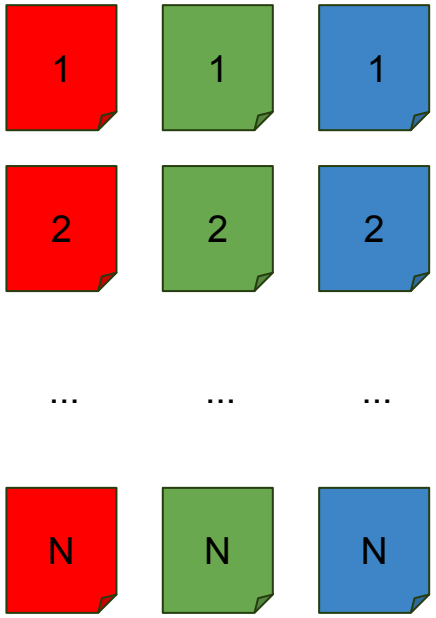


PTero Architecture

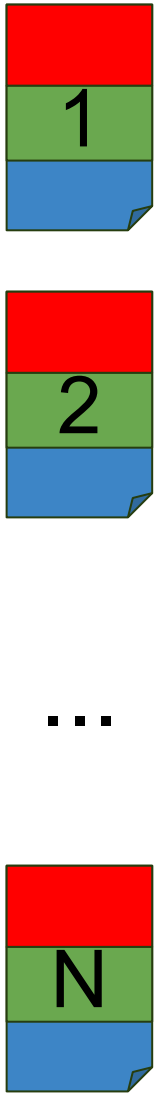
One Sample Per File
Many Chromosomes Per File



One Sample Per File
One Chromosome Per File

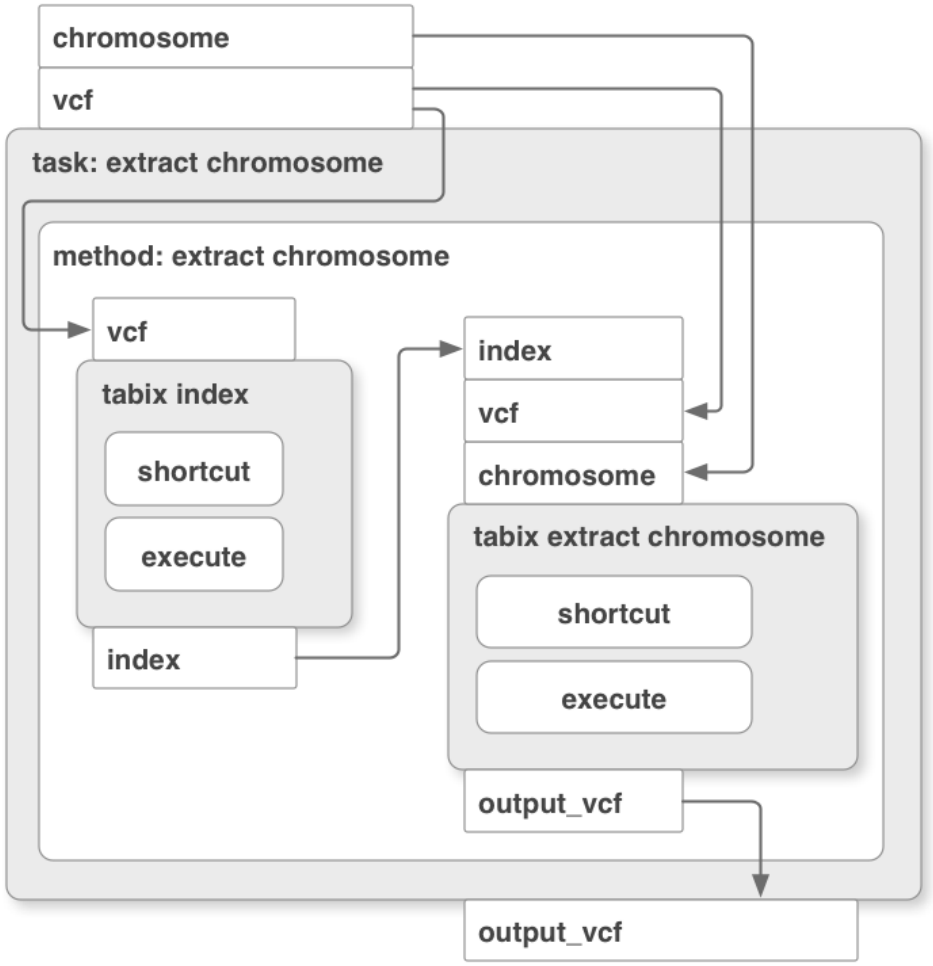


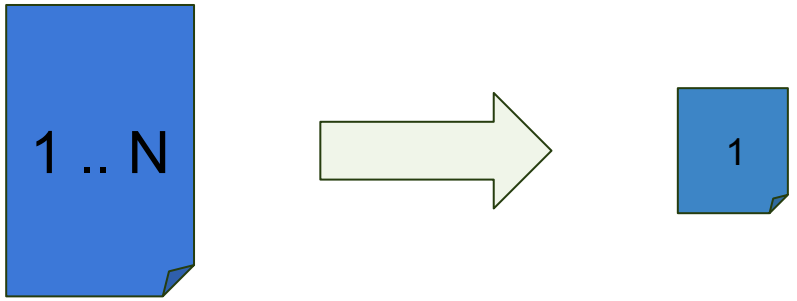
Many Samples Per File
One Chromosome Per File



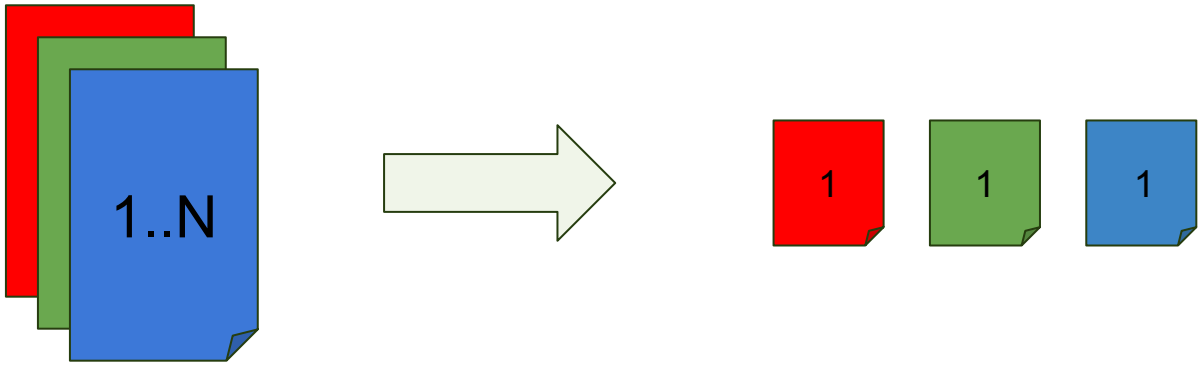
PTero Architecture

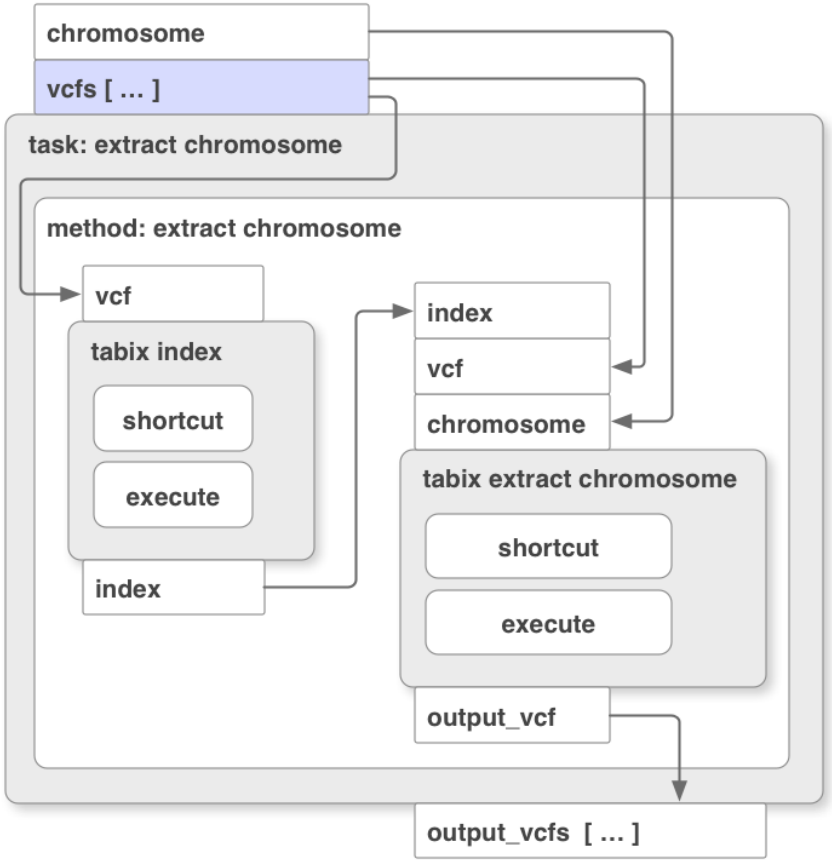
PTero Architecture



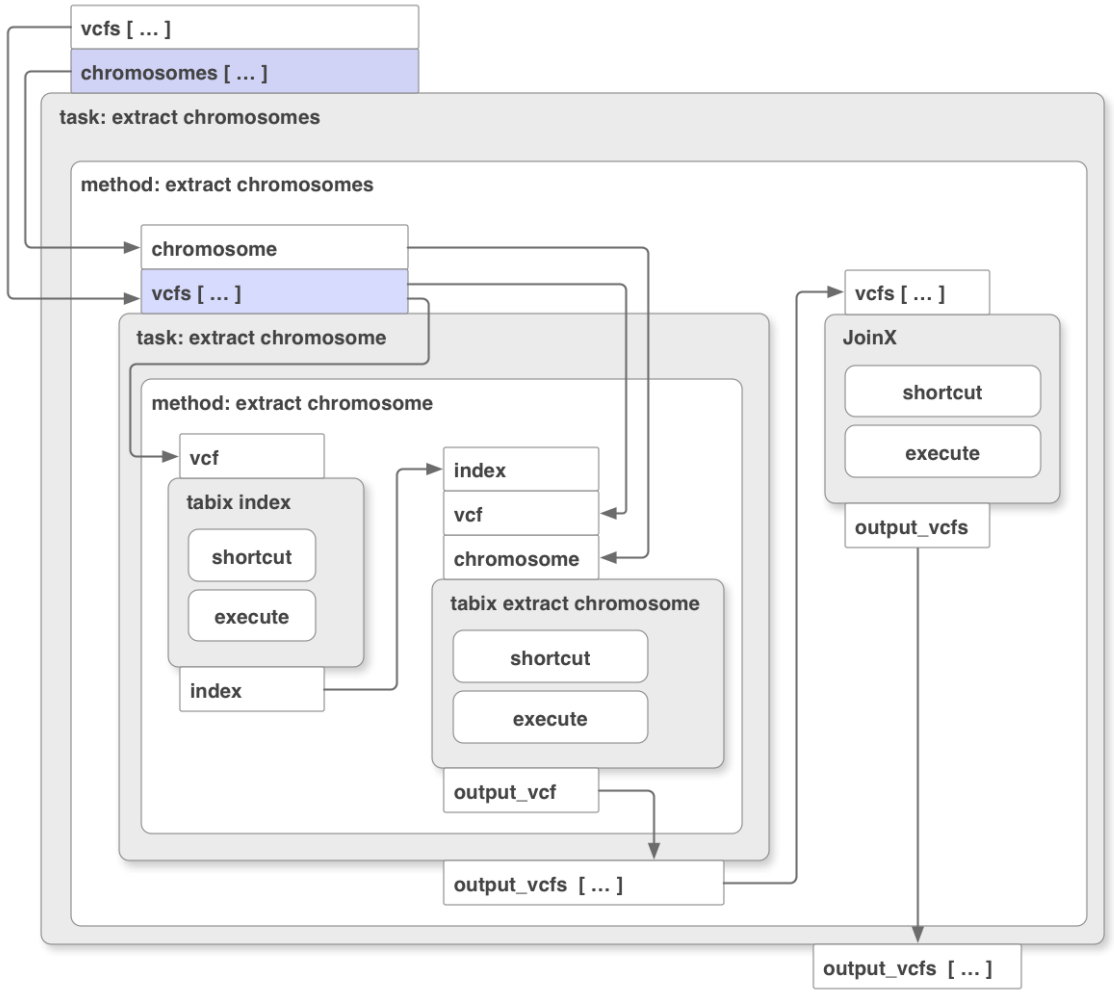


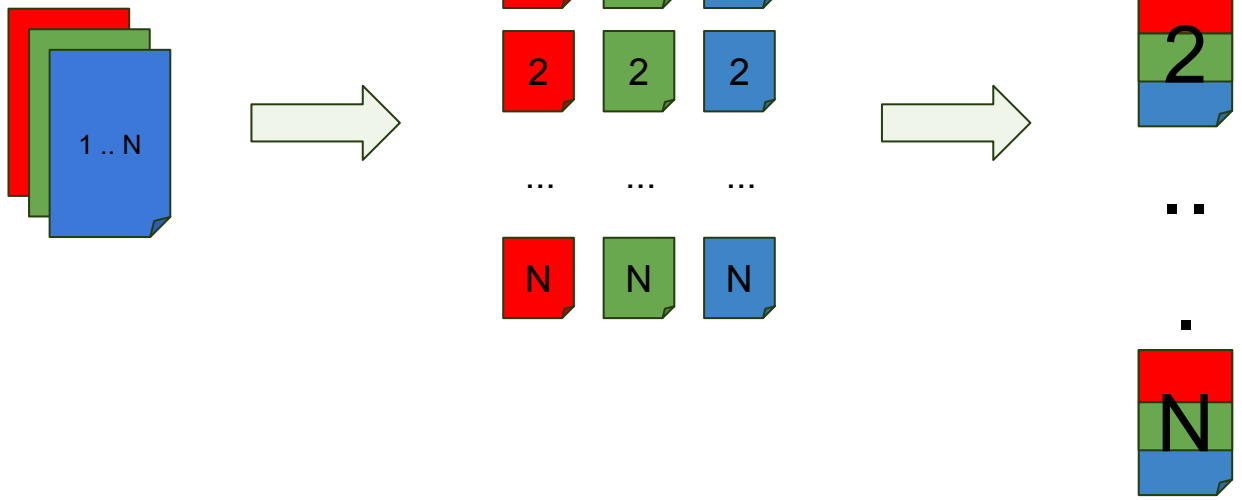
PTero Architecture





PTero Architecture





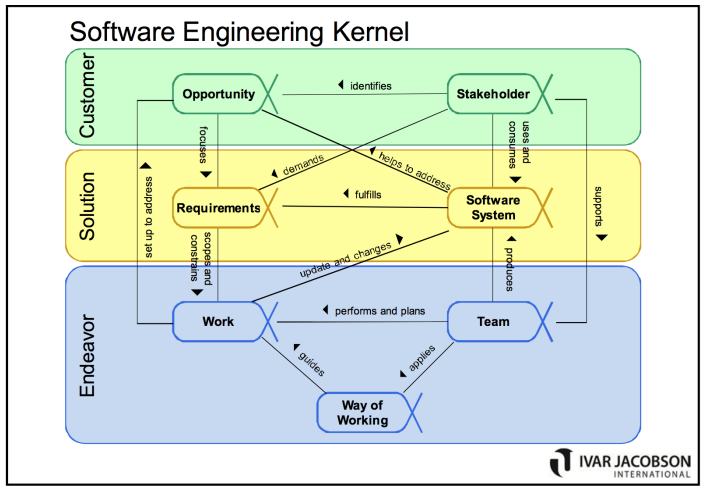
PTero Architecture

```
$ ptero-view http://workflow-service/v1/workflows/5
          TYPE     STATUS              STARTED      DURATION  NAME
      Workflow  succeeded  2015-09-23T23:30:55      00:00:18  FFFJMALcRvKCKoGe7yPTzg
          Task  succeeded  2015-09-23T23:30:57      00:00:16  merge chromosome vcfs [parallel-by: chromosome]
          Task  succeeded  2015-09-23T23:30:57      00:00:16  . merge chromosome vcfs [0]
           DAG  succeeded  2015-09-23T23:30:57      00:00:16  . . merge chromosome vcfs
          Task  succeeded  2015-09-23T23:30:57      00:00:13  . . . extract chromosome from vcf [parallel-by: vcf]
          Task  succeeded  2015-09-23T23:30:57      00:00:12  . . . . extract chromosome from vcf [0, 0]
           DAG  succeeded  2015-09-23T23:30:58      00:00:11  . . . . . extract chromosome from vcf
          Task  succeeded  2015-09-23T23:30:58      00:00:05  . . . . . . index_vcf task
           Job     failed  2015-09-23T23:31:00      00:00:00  . . . . . . . shortcut method
           Job  succeeded  2015-09-23T23:31:01      00:00:02  . . . . . . . index_vcf method
          Task  succeeded  2015-09-23T23:31:03      00:00:06  . . . . . . extract_chromosome task
           Job     failed  2015-09-23T23:31:04      00:00:02  . . . . . . . shortcut method
           Job  succeeded  2015-09-23T23:31:07      00:00:02  . . . . . . . extract_chromosome method
          Task  succeeded  2015-09-23T23:30:57      00:00:13  . . . . extract chromosome from vcf [0, 1]
           DAG  succeeded  2015-09-23T23:30:58      00:00:12  . . . . . extract chromosome from vcf
          Task  succeeded  2015-09-23T23:30:58      00:00:06  . . . . . . index_vcf task
           Job     failed  2015-09-23T23:31:00      00:00:01  . . . . . . . shortcut method
           Job  succeeded  2015-09-23T23:31:03      00:00:01  . . . . . . . index_vcf method
          Task  succeeded  2015-09-23T23:31:04      00:00:05  . . . . . . extract_chromosome task
           Job     failed  2015-09-23T23:31:06      00:00:01  . . . . . . . shortcut method
           Job  succeeded  2015-09-23T23:31:09      00:00:00  . . . . . . . extract_chromosome method
          Task  succeeded  2015-09-23T23:31:10      00:00:02  . . . join_vcfs task
           Job     failed  2015-09-23T23:31:11      00:00:00  . . . . shortcut method
           Job  succeeded  2015-09-23T23:31:12      00:00:00  . . . . join_vcfs method
          Task  succeeded  2015-09-23T23:30:57      00:00:16  . merge chromosome vcfs [1]
           DAG  succeeded  2015-09-23T23:30:57      00:00:16  . . merge chromosome vcfs
          Task  succeeded  2015-09-23T23:30:57      00:00:13  . . . extract chromosome from vcf [parallel-by: vcf]
          Task  succeeded  2015-09-23T23:30:57      00:00:12  . . . . extract chromosome from vcf [1, 0]
           DAG  succeeded  2015-09-23T23:30:58      00:00:11  . . . . . extract chromosome from vcf
          Task  succeeded  2015-09-23T23:30:58      00:00:05  . . . . . . index_vcf task
           Job  succeeded  2015-09-23T23:31:00      00:00:00  . . . . . . . shortcut method
          Task  succeeded  2015-09-23T23:31:03      00:00:06  . . . . . . extract_chromosome task
           Job     failed  2015-09-23T23:31:04      00:00:02  . . . . . . . shortcut method
           Job  succeeded  2015-09-23T23:31:07      00:00:02  . . . . . . . extract_chromosome method
          Task  succeeded  2015-09-23T23:30:57      00:00:13  . . . . extract chromosome from vcf [1, 1]
           DAG  succeeded  2015-09-23T23:30:58      00:00:12  . . . . . extract chromosome from vcf
          Task  succeeded  2015-09-23T23:30:58      00:00:07  . . . . . . index_vcf task
           Job  succeeded  2015-09-23T23:31:00      00:00:01  . . . . . . . shortcut method
          Task  succeeded  2015-09-23T23:31:05      00:00:04  . . . . . . extract_chromosome task
           Job     failed  2015-09-23T23:31:06      00:00:01  . . . . . . . shortcut method
           Job  succeeded  2015-09-23T23:31:09      00:00:00  . . . . . . . extract_chromosome method
          Task  succeeded  2015-09-23T23:31:10      00:00:02  . . . join_vcfs task
           Job     failed  2015-09-23T23:31:11      00:00:00  . . . . shortcut method
           Job  succeeded  2015-09-23T23:31:12      00:00:00  . . . . join_vcfs method
$
```

Development and Testing

Development Methodology:

SEMAT



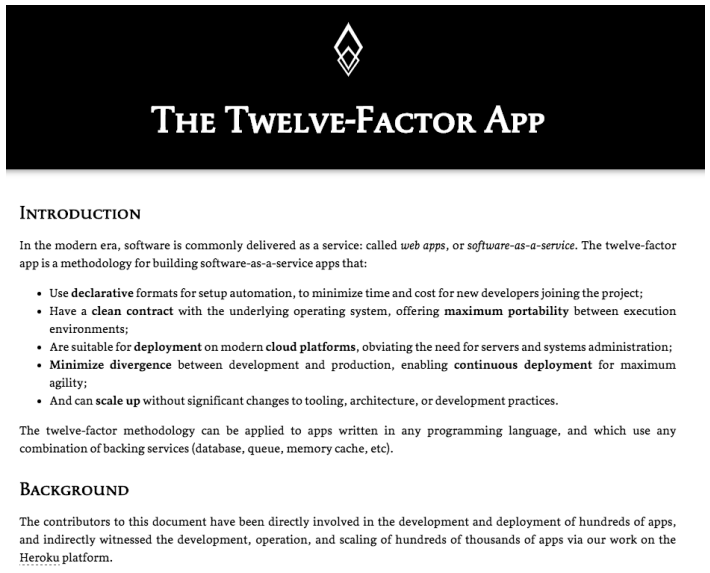
Software Engineering Method and Theory

http://semat.org/

http://www.ivarjacobson.com/semat/

Software System Design:

12 Factor



THE TWELVE-FACTOR APP

INTRODUCTION

In the modern era, software is commonly delivered as a service: called *web apps*, or *software-as-a-service*. The twelve-factor app is a methodology for building software-as-a-service apps that:

- Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;
- Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration;
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility;
- And can **scale up** without significant changes to tooling, architecture, or development practices.

The twelve-factor methodology can be applied to apps written in any programming language, and which use any combination of backing services (database, queue, memory cache, etc).

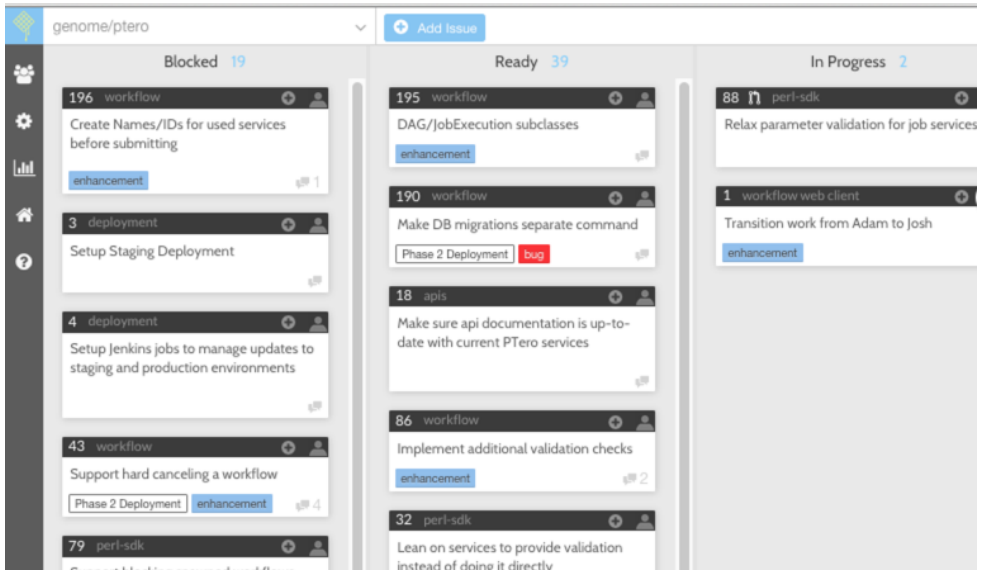
BACKGROUND

The contributors to this document have been directly involved in the development and deployment of hundreds of apps, and indirectly witnessed the development, operation, and scaling of hundreds of thousands of apps via our work on the Heroku platform.

http://12factor.net/

Issue Tracking:

GitHub and waffle.io

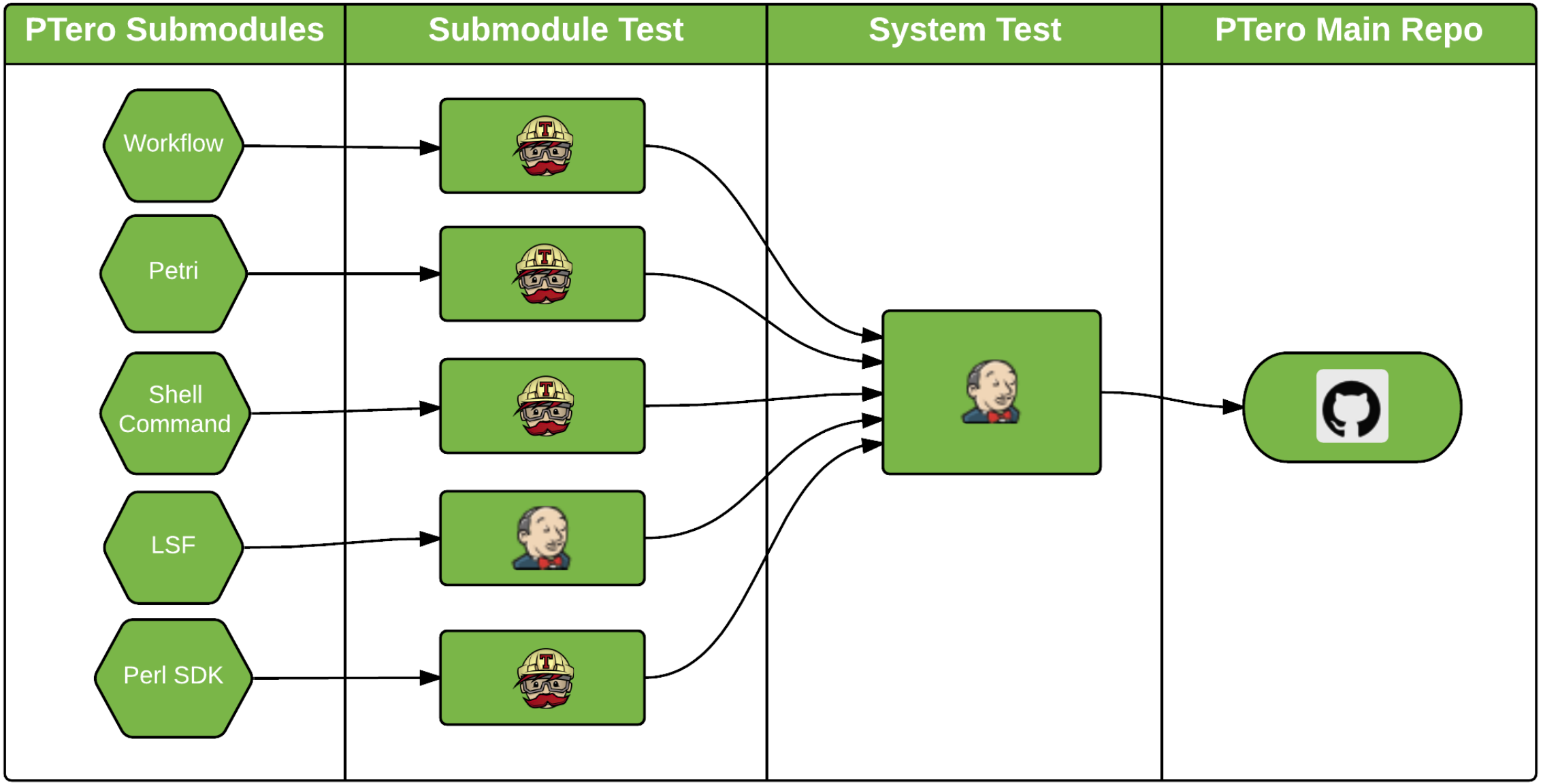


https://waffle.io/

Development

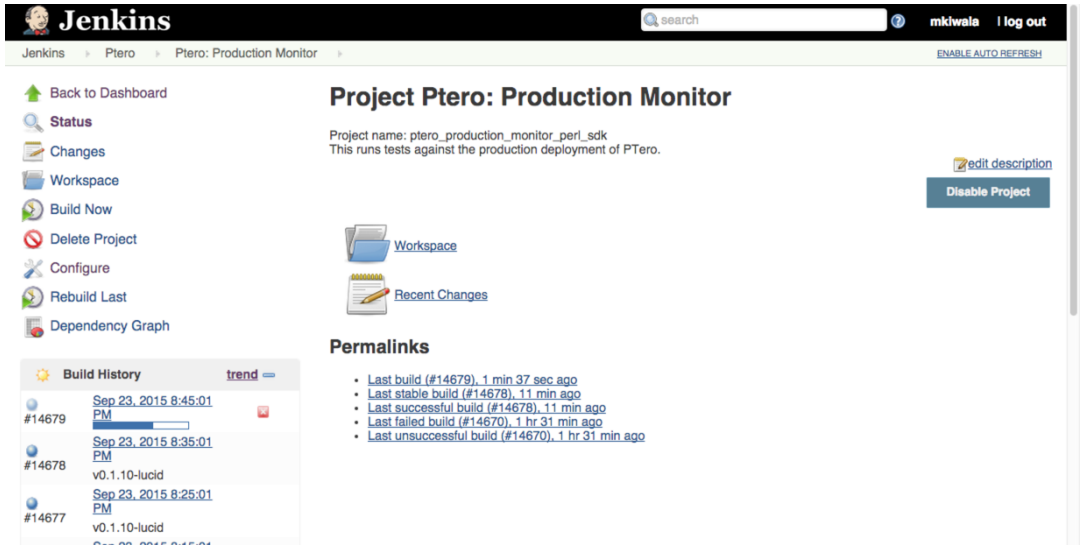
Repo	Language	Sloccount	Testing Sloccount	Test Coverage
Workflow	Python	2,965	1,957	93%
Petri	Python lua	1,765 222	1,873 -	95% -
ShellCommand	Python	291	647	83%
LSF	Python	486	452	76%
PerlSDK	Perl	1767	442	92%

Development and Testing



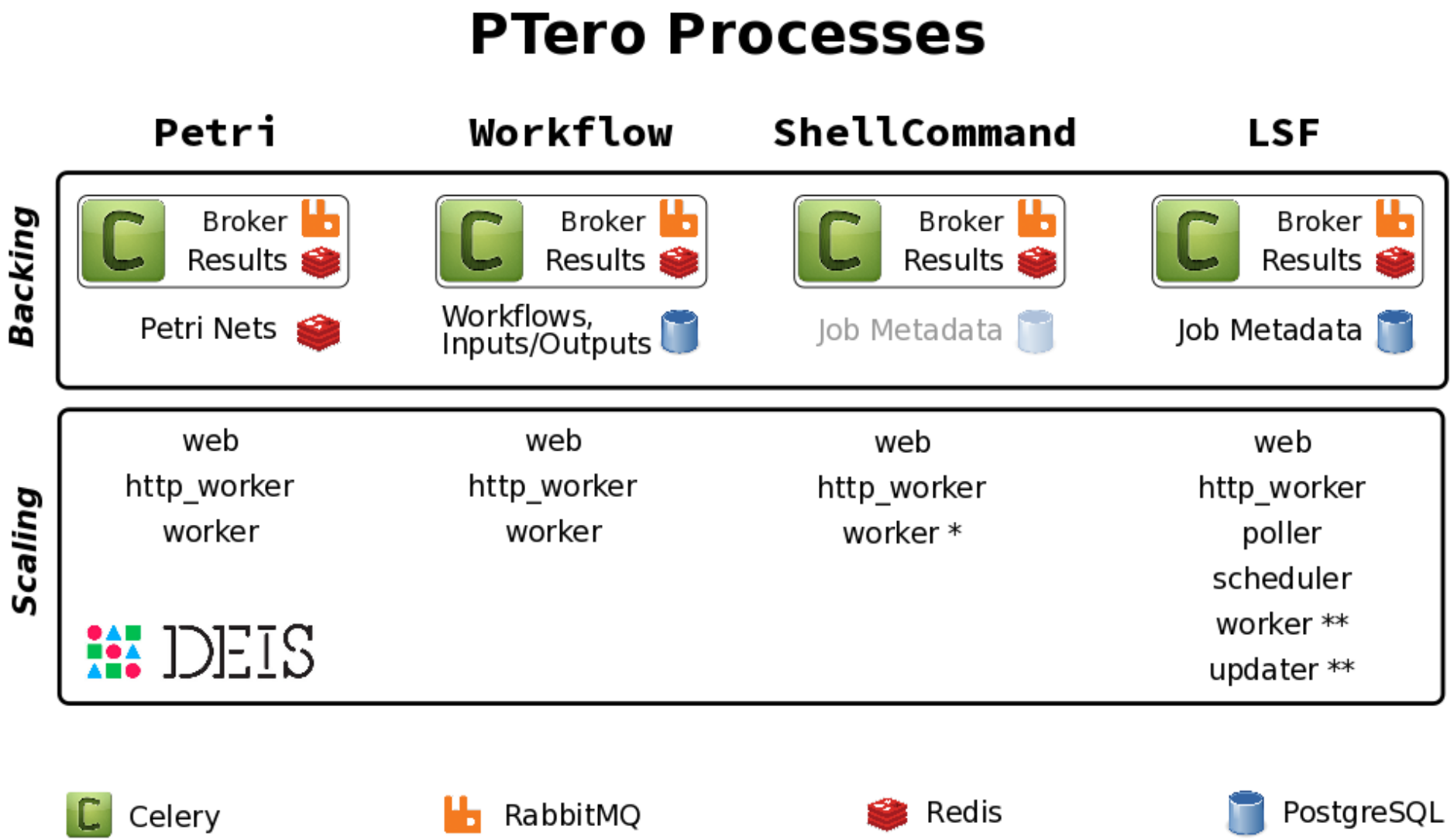
Deployment

System tests as production monitoring



Logstash with Kibana Frontend

Deployment



* Deploy where you want shell-commands to be executed

** Deploy where you have ability to submit and query for LSF jobs (must be LSF submit host)

Development and Testing

```
git clone http://github.com/genome/ptero.git
cd ptero
git submodule update --init
RUN_TESTS=1 vagrant up
```

PTero Architecture

POST to http://ptero-workflow/v1/workflows


```
{
  "inputs": {},
  "links": [
    {
      "destination": "output connector",
      "source": "Say Hello"
    },
    {
      "destination": "Say Hello",
      "source": "input connector"
    }
  ],
  "tasks": {
    "Say Hello": {
      "methods": [
        {
          "name": "print hello",
          "parameters": {
            "commandLine": [
              "bash",
              "-c",
              "echo 'hello'; echo 'world!' 1>&2"
            ],
            "environment": {},
            "user": "vagrant",
            "workingDirectory": "/home/vagrant/ptero/services/workflow/tests/scripts"
          },
          "service": "job",
          "serviceUrl": "http://localhost:5000/v1"
        }
      ]
    }
  }
}
```


- No inputs
- Control-flow links (no data to route)
- 1 Task named “Say Hello”
- 1 Method in that task named “print hello”

Conclusion

PTero is...

- designed to carry out large complex workflows
- service oriented with simple RESTful APIs
- focused in scope, heavily tested, extensible
- free and open source software
- under active development at

github.com/genome/ptero

Acknowledgements

PTero Development Team

Travis Abbott

Mark Burnett

Charlotte Ferguson

Michael Kiwala
mkiwala@wustl.edu

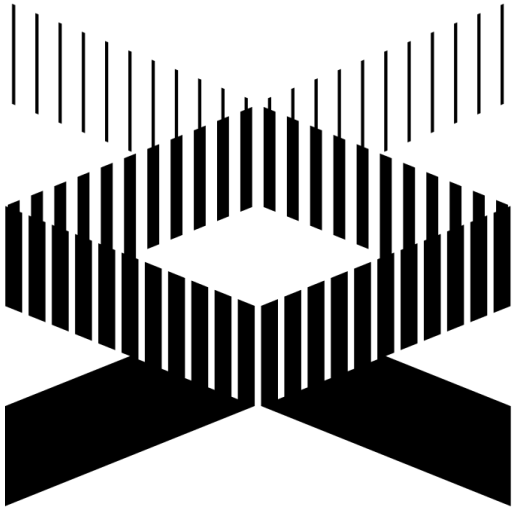
Josh McMichael

David Morton
dmorton@wustl.edu

People

We are the McDonnell Genome Institute at Washington University. Haley Abel, Benjamin Ainscough, Derek Albracht, Mark Ames, Lucinda Fulton, Elizabeth Appelbaum, Rachel Austin, Abhijit Badve, Matthew Bailey, Henry Bauer, Michael Becker, Edward Belter, Angela Bentlage, Jessica Bentlage, Thomas Biewald, Andrew Bohannon, Kirk Brege, Anthony Brummett, Theresa Caesar, Matthew Callaway, Song Cao, Darren Casimere, Wan Ching Chan, Gue Su Chang, Colby Chiang, Young-Jun Choi, Mari Jo Clark, Adam Coffman, Thomas Connell, Lisa Cook, Matthew Cordes, Laura Courtney, Amy D'Albora, John Dalbora, Indraniel Das, Anthony DeLuca, Tracie DeLuca, Sara Decarlo, Brandon Delgado, Ryan Demeter, Brian Derickson, Li Ding, Feiyu Du, James Eldred, Efrem Exum, Candace Farmer, Kelley Foyil, Rachel Frobel, Catrina Fronick, William Fronick, Robert Fugina, Bradley Fulton, Miranda Fulton, Robert Fulton, Kristi Futhey, Eric Geldmacher, Barbara Gillam, Chakravarthy Girda, Jennifer Godfrey, Susan Grasso, Julia Gray, Malachi Griffith, Obi Griffith, Jeanne Grigsby, David Gudermuth, Ira Hall, Terri Hall, Amy Hawkins, Todd Hepler, Jonathan Heusel, Jennifer Hodges, Kuan-lin Huang, Kimberly Hughes-Barnes, Stephanie Jackson, Reyka Jayasinghe, Rodney Jones, Krishna Kanchi, Chul Joo Kang, Kimberly Keen, Shamika Ketkar, Kyung Kim, Michael Kiwala, Daniel Koboldt, Sara Kohlberg, Colin Kremitzki, Milinn Kremitzki, Scott Kruchowski, Kilannin Krysiak, Jason Kunisaki, Nathan Kunkee, Sheila Lakanen, David Larson, Shawn Leonard, Shin Leong, Robert Lesurf, Andrew Levy, Shannon Lewis, Timothy Ley, Tiandao Li, Tina Lindsay, Charles Lu, Amy Ly, Sandra MacMillan, Vincent Magrini, Lenon Maguire, Christopher Maher, Jennifer Maher, Elaine Mardis, Christopher Markovic, John Martin, Robert Mashl, Judith McCart, Sean McGrath, Michael McLellan, Joshua McMichael, Samantha McNulty, Brian Meininger, Karyn Meltz Steinberg, Kathie Mihindukulasuriya, Christopher Miller, Jordan Minx, Patrick Minx, Makedonka Mitreva, Deborah Moeller, Michael Montague, Thomas Mooney, David Morton, Pamela Nangle, William Nash, Joanne Nelsonkary, Ricky Ng, Mike Nhan, Thomas Nicholas, Jie Ning, David O'Brien, Gretchen O'Donnell, Michelle O'Laughlin, Kerri Ochoa, Bradley Ozenberger, Philip Ozersky, Stephanie Parrish, Josh Peck, Kymberlie Pepin, Allegra Petti, Craig Pohl, Robert Pokorney, Eric Ponce, Avinash Ramu, Allison Regier, Amy Reily, Susan Rock, Irina Ronko, Bruce Rosa, Kelsi Rotter, Ryan Rupp, Amy Sansone, Kyriena Schatzkamer, Debra Scheer, Heather Schmidt, Michael Schmidt, William Schroeder, Adam Scott, Sohini Sengupta, Cherilynn Shadding, Nicholas Sheehan, Susanna Siebert, Zachary Skidmore, Julie Smith, Steven Spargur, Nicholas Spies, Craig Steinmeyer, Scott Stevens, Bruce Striegel, Cynthia Strong, Dawn Sutter, Kenneth Swanson, Andrea Taylor, Thynn Thane, Brenda Theising, Brianne Thomeczek, Valerie Thompson, Chad Tomlinson, Lee Trani, Nikki Trapp, Evanne Trevaskis, Richard Trippeer, Rusudan Turabelidze, Rahul Tyagi, Joelle Veizer, Tammi Vickery, Alex Wagner, Jason Waligorski, Jason Walker, Latricia Wallace, Grace Wang, Jiayin Wang, Wesley Warren, Brian Watts, James Weible, Matthew Weil, Michael Wendl, Brian White, Richard K. Wilson, Roxanne Wilson, Aye Wollam, Dee Wu, Song Wu, Matthew Wyczalkowski, Mingchao Xie, Kai Ye,Venkata Yellapantula, Xu Zhang.

Funding



National Human Genome Research Institute

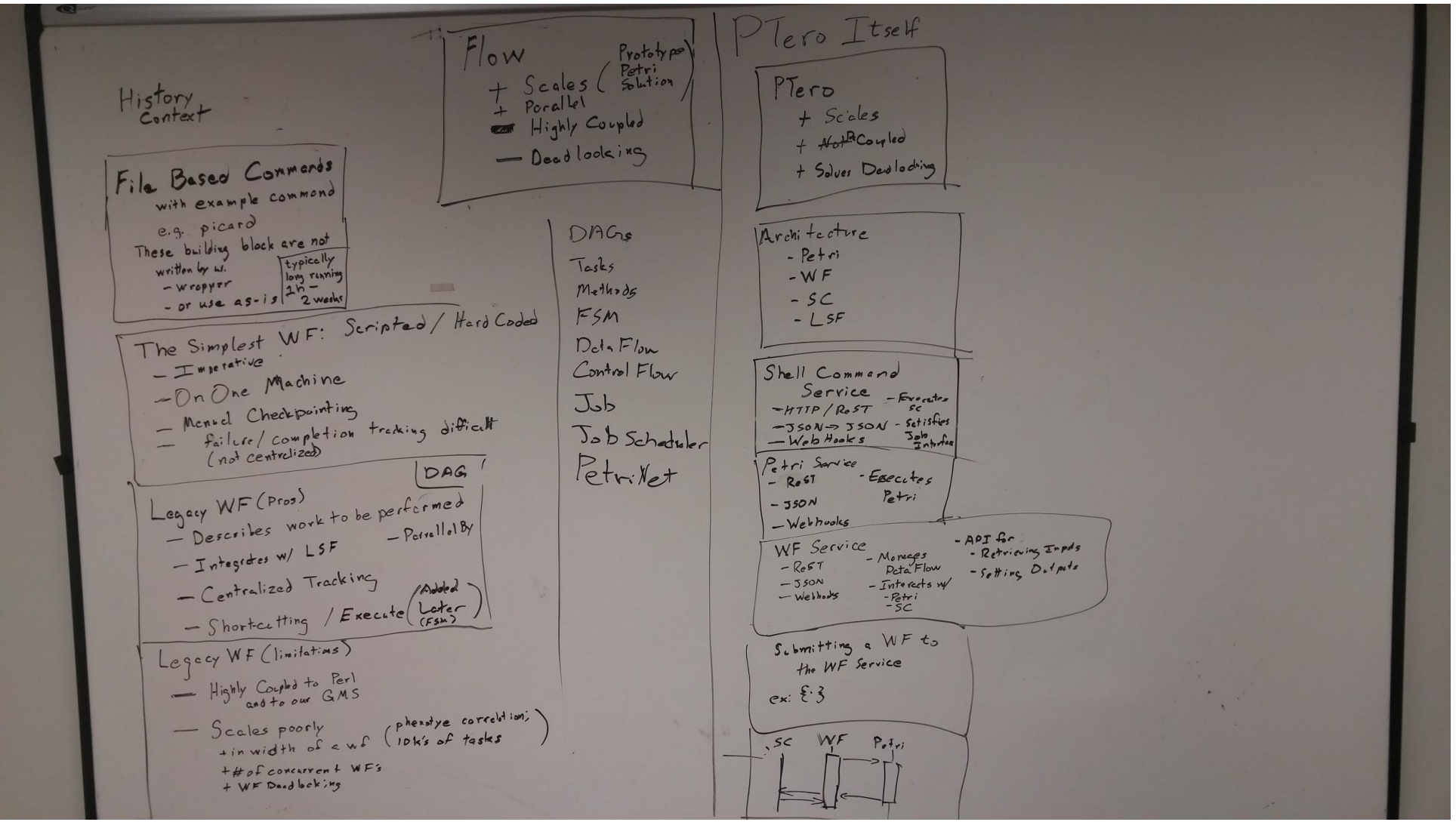
Grant: U54 HG003079 (Dr Wilson)

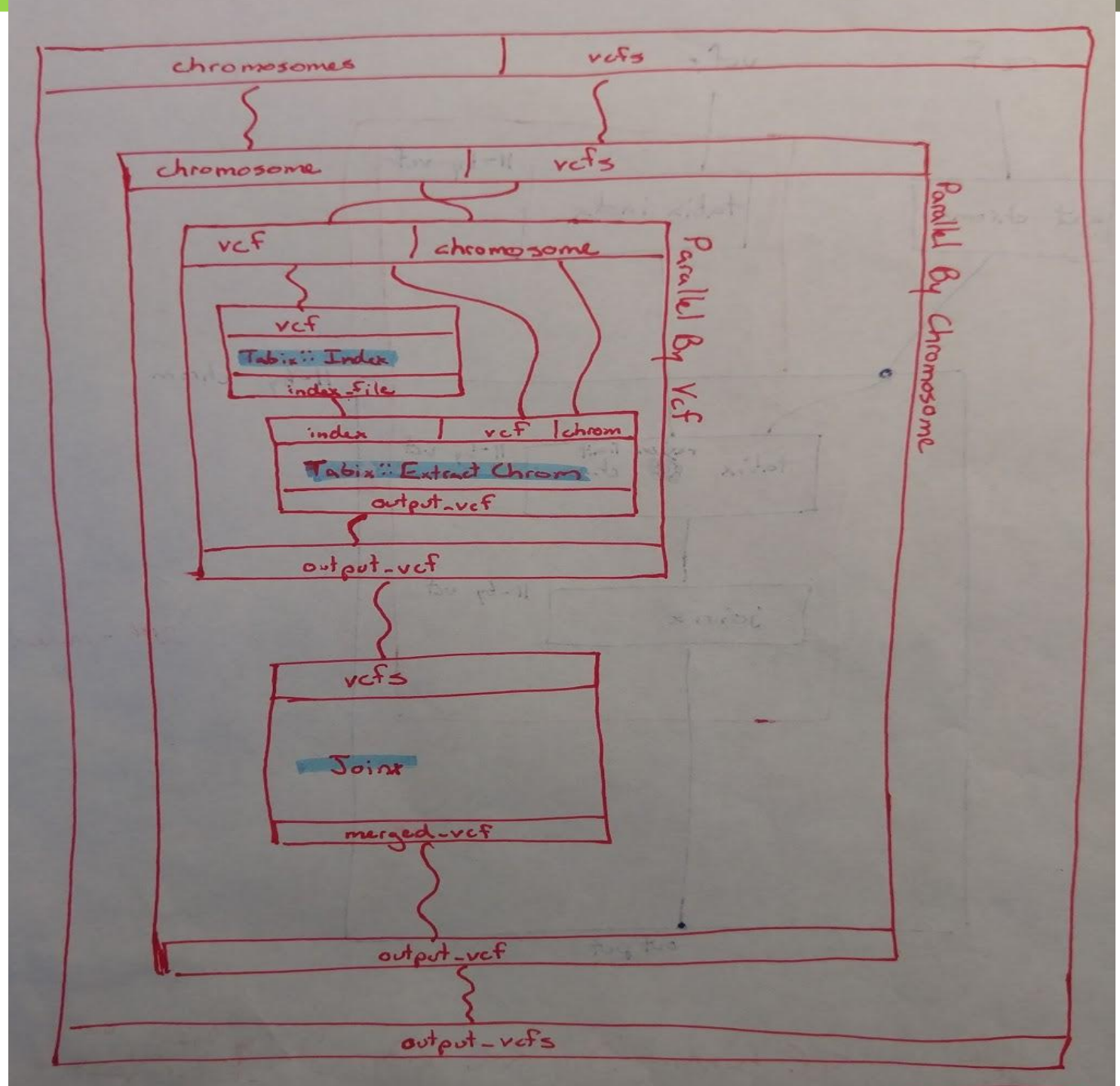
Technologies

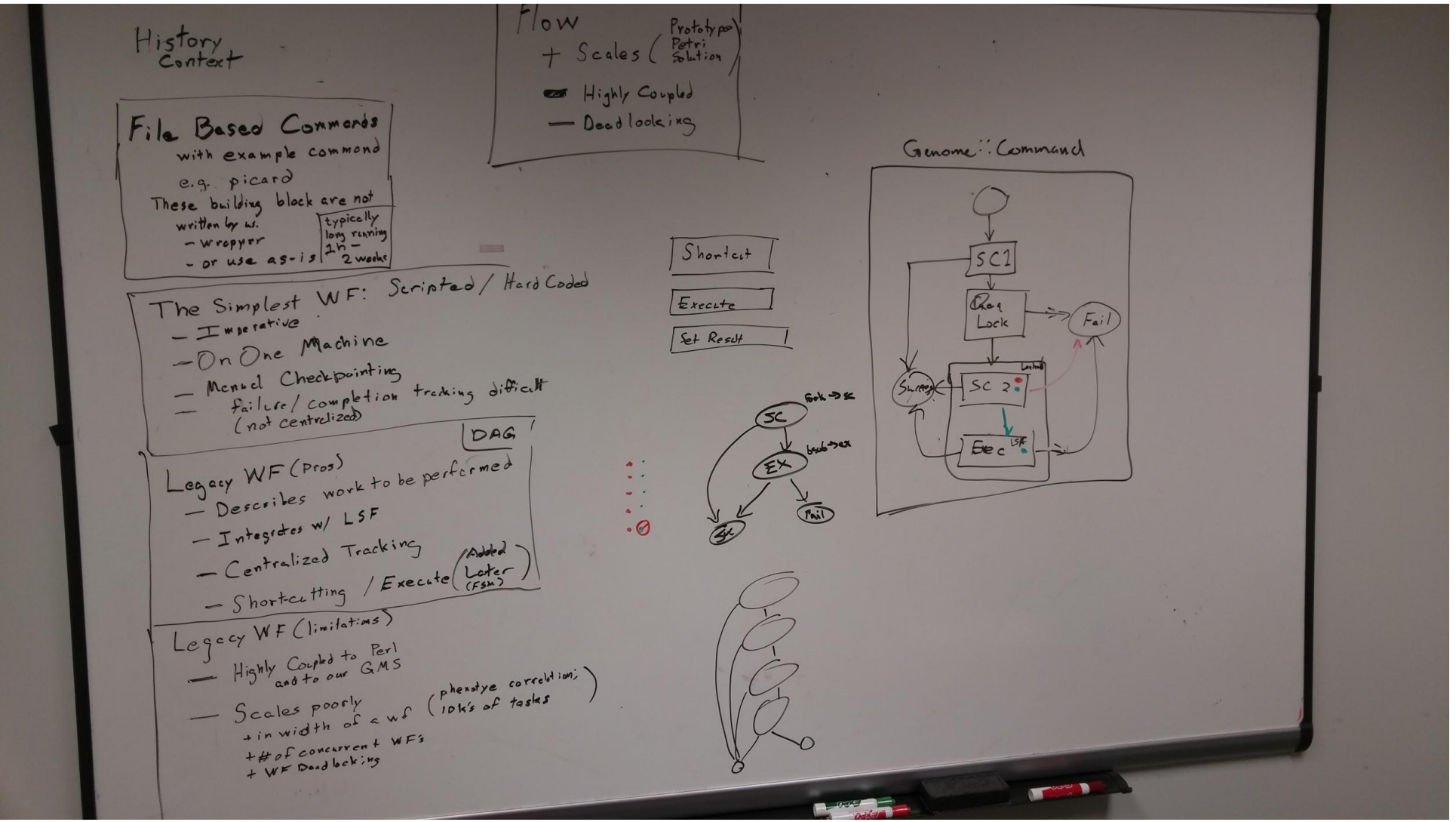
Postgresql
Redis
RabbitMQ
Python
Ubuntu
CoreOS
Celery
Sqlalchemy
Alembic

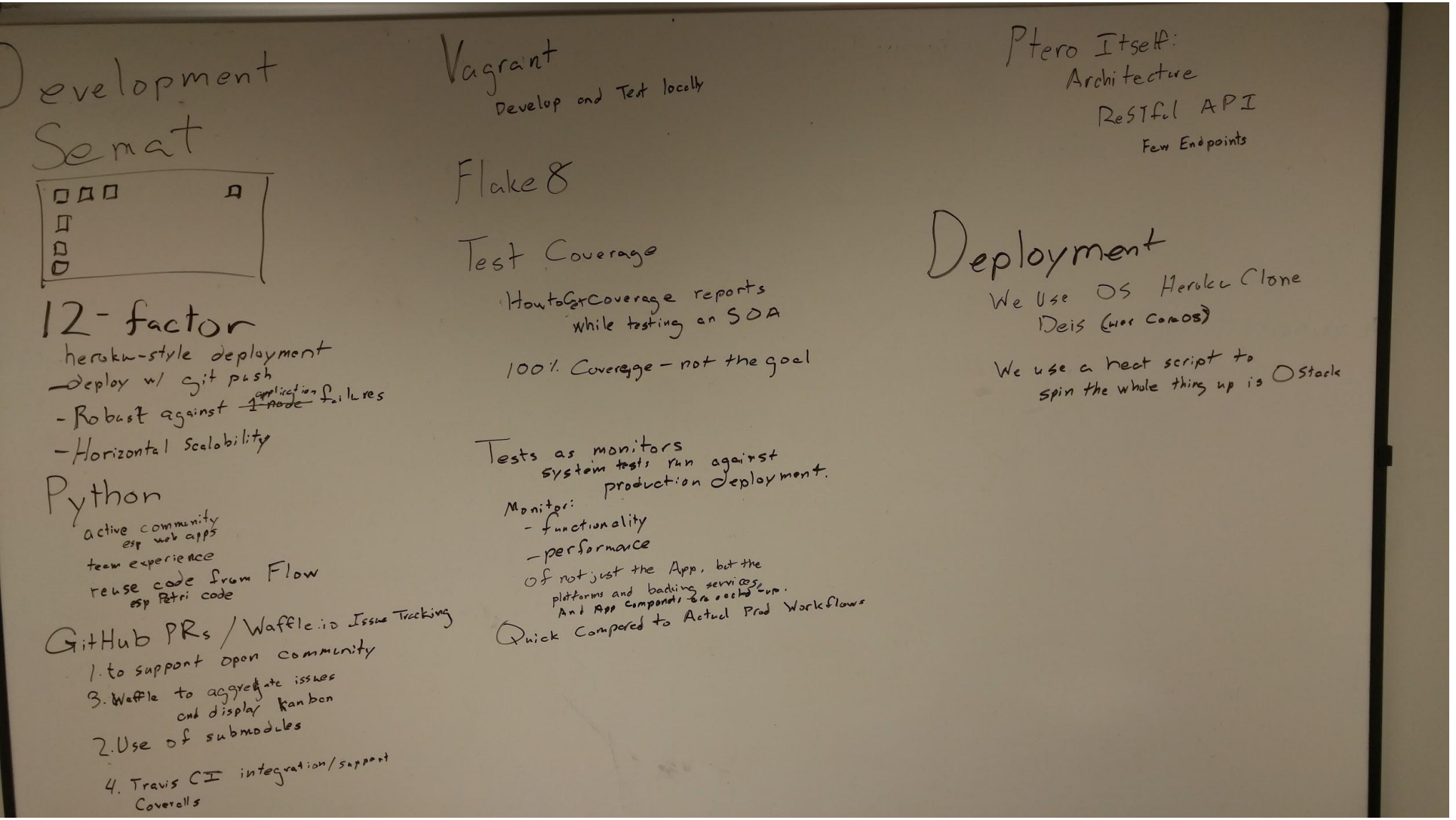
Flask
Perl5
tox
Travis CI
gunicorn
honcho
jsonschema
Moose

Deis
vmWare
OpenStack
Heat Orchestration Template
VirtualBox
Docker


History
Context
Flow
+ Scales (Prototype Petri Solution)
+ Parallel
Highly Coupled
— Deadlocking
PTero Itself
PTero
+ Scales
+ Coupled
+ Solves Deadlocking
File Based Commands
with example command
e.g. picard
These building block are not written by us.
- wrapper
- or use as-is
typically long running 1h – 2 weeks
DAGs
Tasks
Methods
FSM
Data Flow
Control Flow
Job
Job Scheduler
Petri Net
Architecture
- Petri
- WF
- SC
- LSF
Shell Command Service
- HTTP / ReST
- JSON → JSON
- Web Hooks
- Executes SC
- Satisfies Job Interface
Petri Service
- ReST
- JSON
- Web hooks
- Executes Petri
The Simplest WF: Scripted / Hard Coded
— Imperative
— On One Machine
— Manual Checkpointing
— failure / completion tracking difficult (not centralized)
DAG
Legacy WF (Pros)
— Describes work to be performed
— Integrates w/ LSF
— ParrallelBy
— Centralized Tracking
— Shortcutting / Execute (Added Later (FSM))
WF Service
- ReST
- JSON
- Webhooks
- Manages Data Flow
- Interacts w/ Petri
- SC
- API for
- Retrieving Inputs
- Setting Outputs
Legacy WF (limitations)
— Highly Coupled to Perl and to our GMS
— Scales poorly (phenotye correlation; 10k's of tasks)
+ in width of a wf
+ # of concurrent WFs
+ WF Deadlocking
Submitting a WF to the WF Service
ex: E.3
SC
WF
Petri


chromosomes
vcfs
chromosome
vcfs
Parallel By Chromosome
vcf
chromosome
Parallel By Vcf
vcf
Tabix:: Index
index_file
index
vcf
chrom
Tabix:: Extract Chrom
output_vcf
output_vcf
vcfs
Joint
merged_vcf
output_vcf
output_vcfs


History
Context
Flow
+ Scales (Prototype Petri Solution)
— Highly Coupled
— Deadlocking
File Based Commands
with example command
e.g. picard
These building block are not written by us.
- wrapper
- or use as-is
typically long running 1h – 2 weeks
Genome::Command
SC1
Req Lock
Fail
SC 2
Exec
LSF
Shortcut
Execute
Set Result
The Simplest WF: Scripted / Hard Coded
— Imperative
— On One Machine
— Manual Checkpointing
— failure/completion tracking difficult (not centralized)
DAG
SC
EX
Fail
Legacy WF (Pros)
— Describes work to be performed
— Integrates w/ LSF
— Centralized Tracking
— Shortcutting / Execute (Added Later (FSM))
Legacy WF (limitations)
— Highly Coupled to Perl and to our GMS
— Scales poorly (phenotype correlations; 10k's of tasks)
+ in width of a wf
+ # of concurrent WF's
+ WF Deadlocking

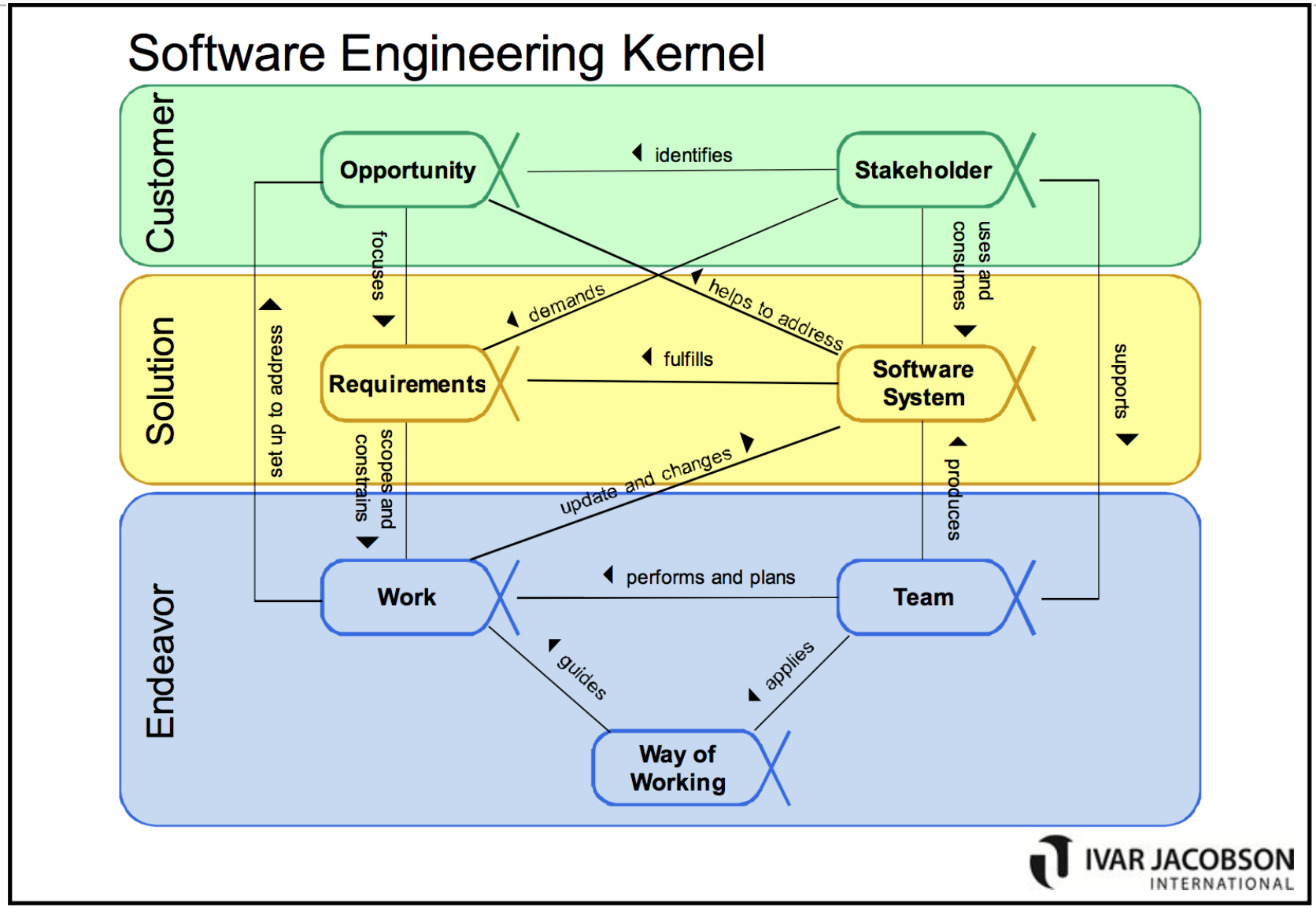

Development
Semat
12-factor
heroku-style deployment
-Deploy w/ git push
-Robust against ~~1 node~~ application failures
-Horizontal Scalability
Python
active community
esp web apps
team experience
reuse code from Flow
esp Petri code
GitHub PRs / Waffle.io Issue Tracking
1. to support open community
3. Waffle to aggregate issues
and display kanban
2. Use of submodules
4. Travis CI integration/support
Coveralls
Vagrant
Develop and Test locally
Flake8
Test Coverage
HowtoGetCoverage reports
while testing an SOA
100% Coverage - not the goal
Tests as monitors
system tests run against
production deployment.
Monitor:
- functionality
- performance
of not just the App, but the
platforms and backing services
And App components are cached up.
Quick Compared to Actual Prod Workflows
Ptero Itself:
Architecture
RESTful API
Few Endpoints
Deployment
We Use OS Heroku Clone
Deis (uses CoreOS)
We use a heat script to
spin the whole thing up in OStack

Development Practices and Testing Choices

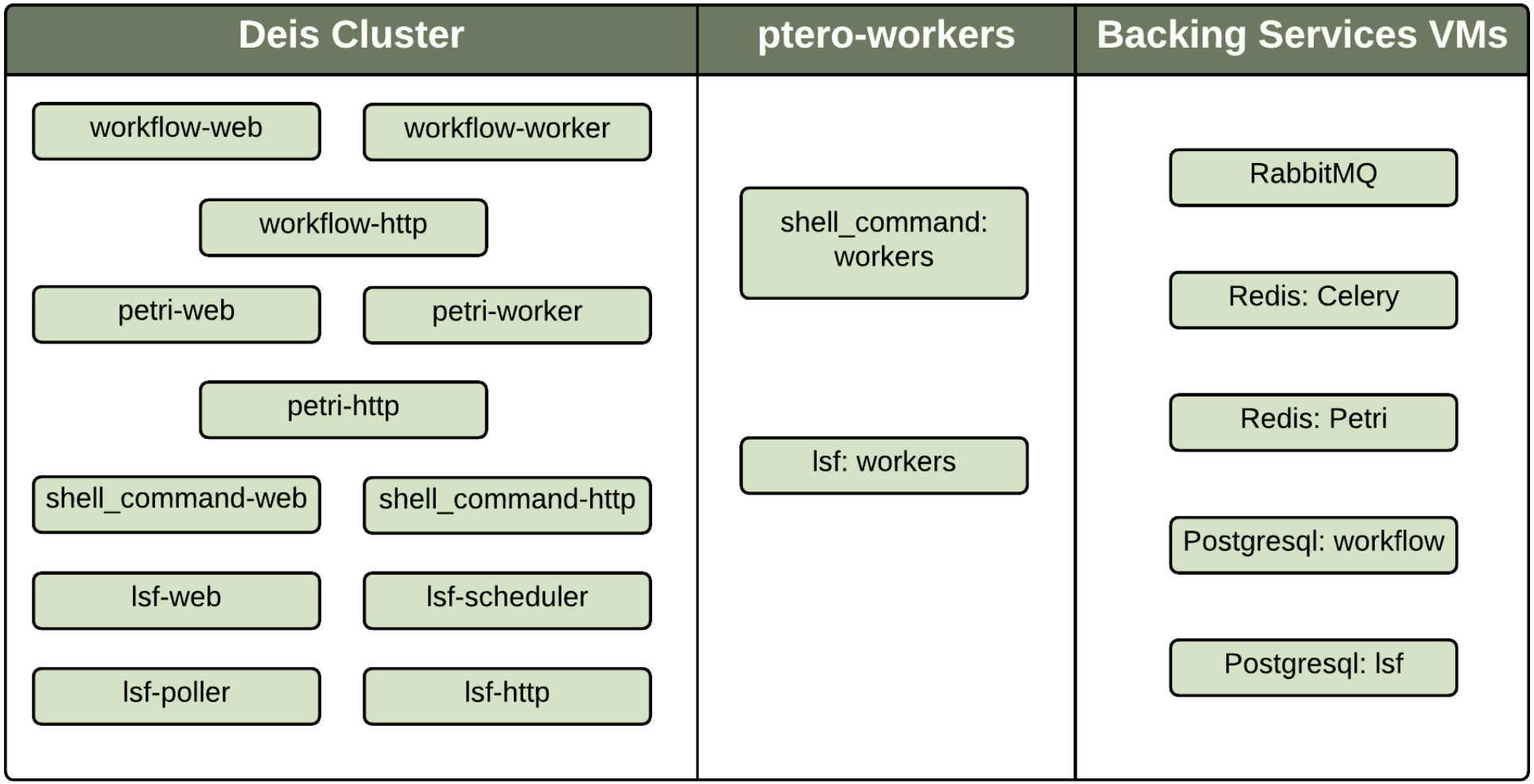
SEMAT

(Software Engineering Method and Theory)

- semat.org
- www.ivarjacobson.com/semat



Deployment


Deis Cluster
workflow-web
workflow-worker
workflow-http
petri-web
petri-worker
petri-http
shell_command-web
shell_command-http
lsf-web
lsf-scheduler
lsf-poller
lsf-http
ptero-workers
shell_command: workers
lsf: workers
Backing Services VMs
RabbitMQ
Redis: Celery
Redis: Petri
Postgresql: workflow
Postgresql: lsf

PTero Architecture

GET http://ptero-workflow/v1/workflows/1

-or-

GET http://ptero-workflow/v1/workflows?name=3SYTDXt4Sp2eL4rtTdxjPg


```
{
  "name": "3SYTDXt4Sp2eL4rtTdxjPg",
  "reports": {
    "workflow-details": "http://localhost:7000/v1/reports/workflow-details?workflow_id=1",
    "workflow-executions": "http://localhost:7000/v1/reports/workflow-executions?workflow_id=1",
    "workflow-outputs": "http://localhost:7000/v1/reports/workflow-outputs?workflow_id=1",
    "workflow-skeleton": "http://localhost:7000/v1/reports/workflow-skeleton?workflow_id=1",
    "workflow-status": "http://localhost:7000/v1/reports/workflow-status?workflow_id=1",
    "workflow-submission-data": "http://localhost:7000/v1/reports/workflow-submission-data?workflow_id=1"
  },
  "status": "succeeded"
}
```


Workflow gets named automatically if one isn't specified at submission time.

Development Practices and Testing Choices