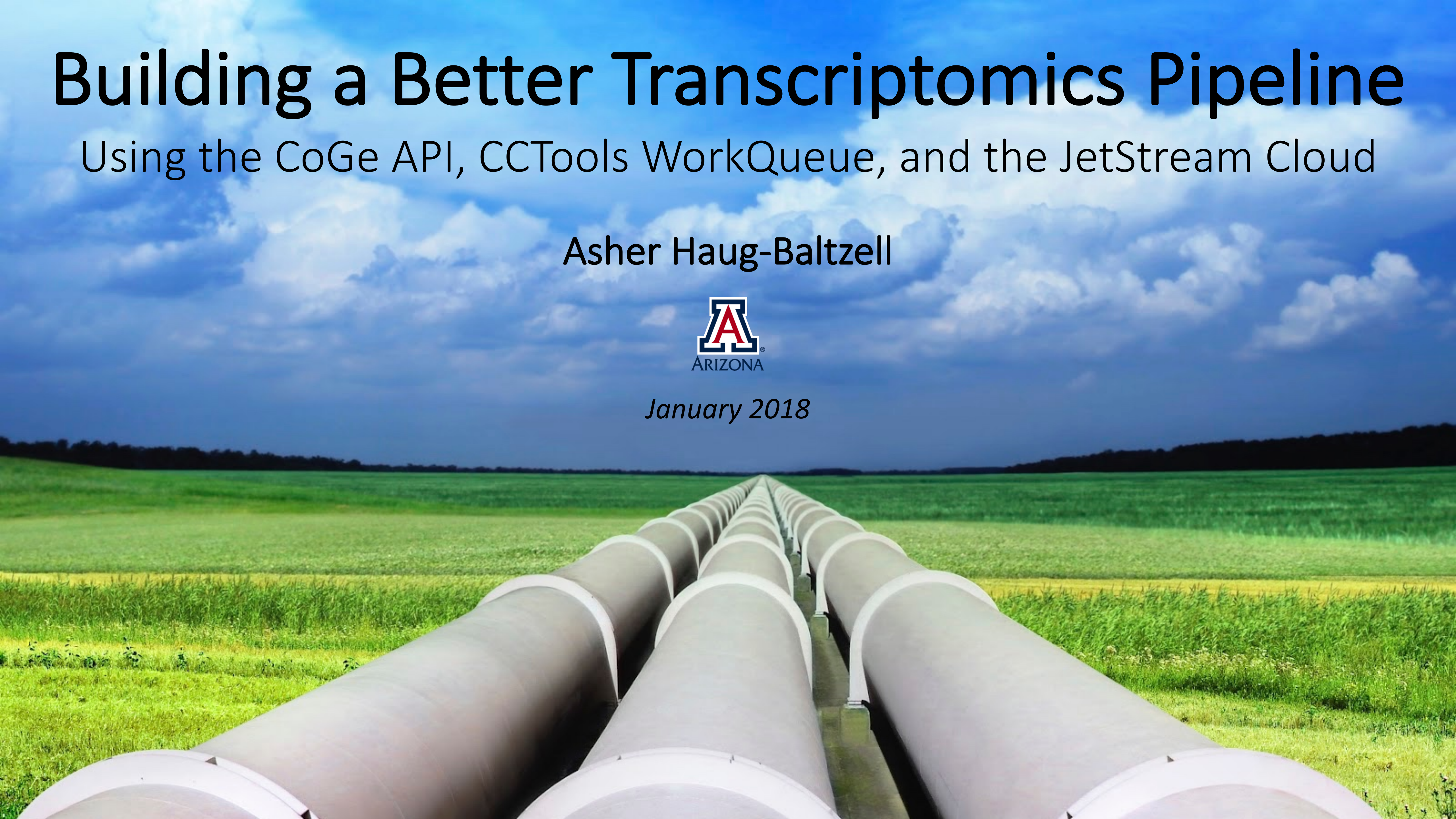
Building a Better Transcriptomics Pipeline

Using the CoGe API, CCTools WorkQueue, and the JetStream Cloud

Asher Haug-Baltzell

Arizona

January 2018

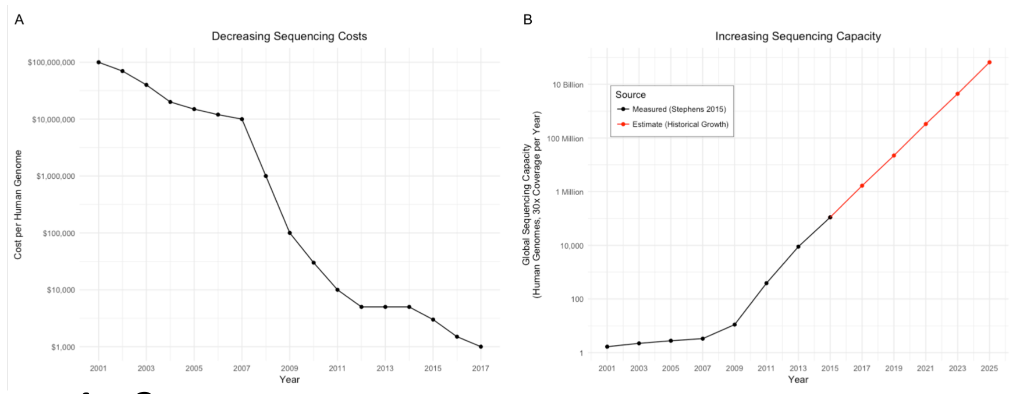


The Premise

*We want to re-analyze **hundreds** to **thousands** of existing RNA-seq datasets.*

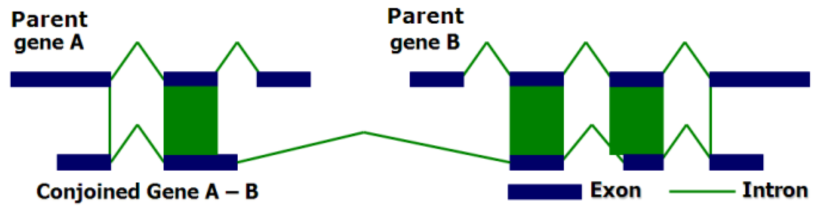
Why?

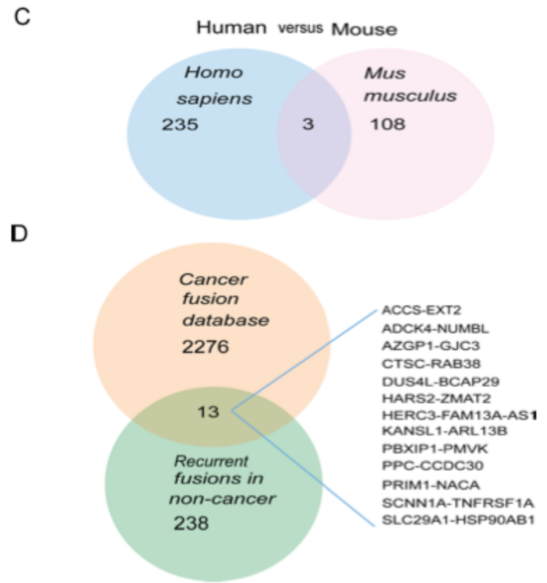
- **Existing datasets** can answer **new questions**
  - RNA-seq provides deep look into transcribed genome.
  - Decreasing costs of sequencing = many datasets generated



Research Question

- ***Are non-canonical transcripts evolutionarily conserved in plant species?***
  - I.e. is "junk" actually junk?
  - Conserved behavior is likely NOT junk.
- "Non Canonical Transcripts"
  - Chimeric transcripts (CT)
  - Read-through transcripts (RT)
- Chimeric transcripts common in mammals:
  - Cancer genomes – possible biomarkers (Zhou et al. 2012)
  - Non-cancer genomes – roles unknown (Babiceanu et al. 2016)

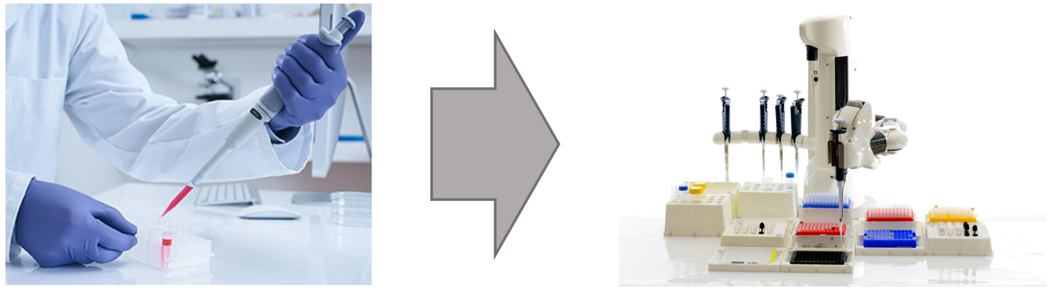




(1) Babiceanu, Mihaela et al. "Recurrent Chimeric Fusion RNAs in Non-Cancer Tissues and Cells." *Nucleic Acids Research*, February, 2016. (2) Zhou, Jianhua, et al. "Chimeric RNAs as Potential Biomarkers for Tumor Diagnosis." *BMB Reports,* March, 2012.

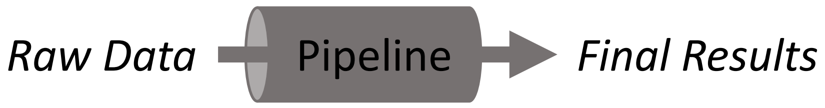
The Premise

*We want to **efficiently** and **automatically** execute these analyses.*



The Pipeline

- **“Pipeline”** - multiple programs linked into single workflow.



- **Goal:** Automate complex tasks.
  1. Pre-processing: QC, raw data processing
  2. Analysis: e.g. variant calling, quantifications, filtering
  3. Post-processing: Format manipulations, visualizations.

Why Build Pipelines?

1. **Easier to Use**
   - Mask complicated parameters/options
   - Quickly rerun if (when):
     - Errors identified
     - New data available
   - Simpler to package & distribute
2. **Less Error Prone**
   - Simplifies otherwise complex tasks
   - Removes “human element”

The Problem

*No **single platform** exists to address all of our needs.*

Analysis Software

- Multiple options for chimeric transcript discovery.
- None directly applicable to plants (polyploidy…)
- Don’t scale well to hundreds/thousands of samples.
- Installation & usage challenges = ↓ reproducibility

Data Management

- Data spread across multiple different platforms
  - Reference Genomes @ EnsemblPlants
  - Reference Genome Info @ CoGe
  - RNAseq @ NCBI SRA
- Practical storage limitations prevent co-locating all data.

Scalable Execution

- Automated.
- Robust.
- Flexible.
- Scalable.
- Portable.

Downstream Analysis

- Custom analysis of final results.
- Portable, common formats.
- Additional result annotations.
- Visualizations.
- Sharable.

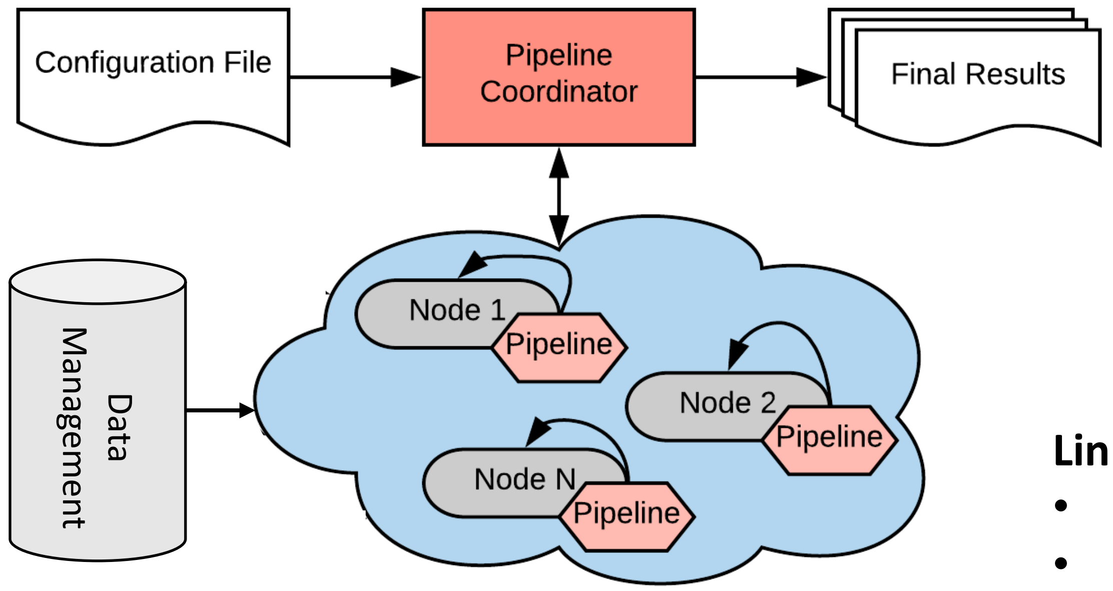
Distribution

- Transparent: Open-source
- Reproducible
  - Packaged
  - Documented

The Solution

*Using bioinformatic resources and platforms, **build a better transcriptomics pipeline**.*

Easy Setup & Execution
- Standardized Config File



Useful Results
- CSVs
- BEDs

Link Parts & Pieces
- Logic: *Python*
- Links: *CCTools WorkQueue*

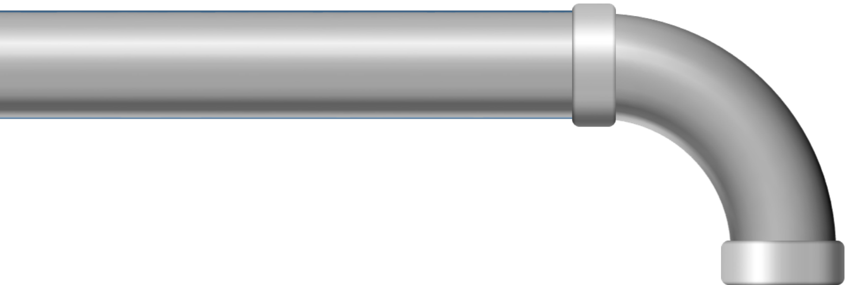
Data Management
- RNASeq Data: *NCBI SRA Toolkit*
- Genome Info: *CoGe API*
- Genome Sequences: *Ensembl*

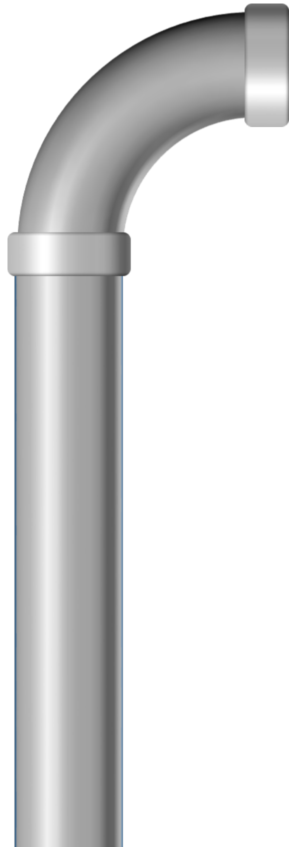
Scalable Data Analysis Resources
- Easy setup: *Docker*
- Scalable Resource: *JetStream Cloud*

- ✓ Automated.
- ✓ Robust.
- ✓ Flexible.
- ✓ Scalable.
- ✓ Portable.

Building a Better Transcriptomics Pipeline

1. Locate, Test & Extend Appropriate Analysis Software
2. Pipeline & Package Analysis Steps
3. Benchmark the Pipeline
4. Build a Coordinating Master Script
5. Identify & Obtain Appropriate Compute Resources
6. Run Analyses, Collect Results
7. Distribute!





Step 1: Locate, Test & Extend Software

Locating & Testing Appropriate Software

- Locate
  - Peer Review - Search for published reviews/comparisons.
  - Self Review – Test multiple options with own data
- Testing
  - Is this software appropriate for my data?
  - Are the results analytically valid?
  - Are any modifications needed? -> **Extend!**

The CoGe API

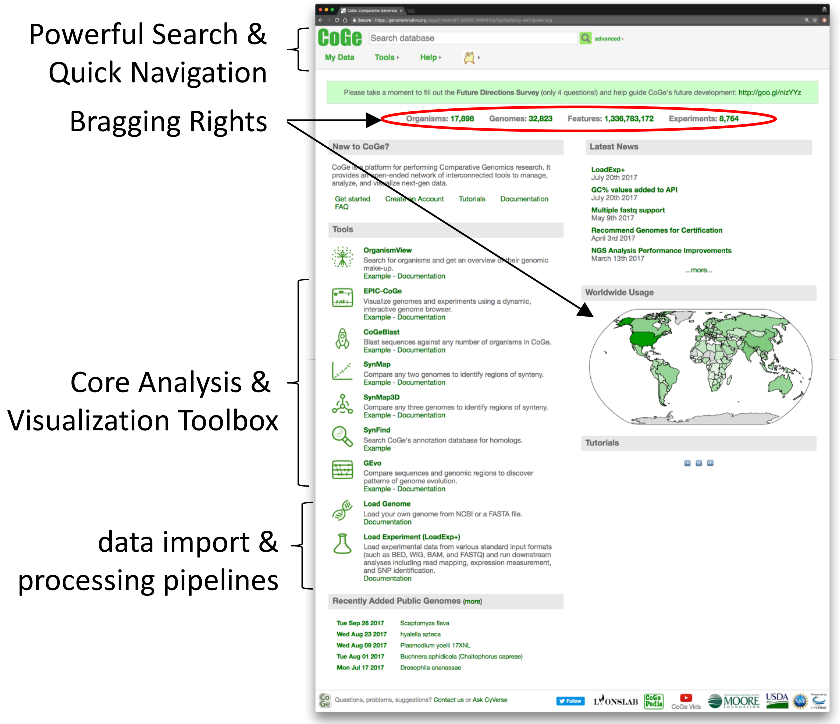
A RESTful API for accessing CoGe databases & analytical pipelines

https://genomevolution.org/wiki/index.php/Web_Services_REST_API

- Search, Fetch, Add, Update, Delete…
  - Organisms
  - Genomes
  - Features
  - Experiments
  - Notebooks
  - Groups Jobs
- Obtain JBrowse-compliant data.

Extending Software

- Not all programs can apply to all organisms.
  - *E.g. Can a human-centric analysis program be modified to work with more complex plant genome data?*
- Not all programs have robust data management.
  - *E.g. If data can't be stored data, can program be modified to allow remote access?*



Powerful Search & Quick Navigation
Bragging Rights
Core Analysis & Visualization Toolbox
data import & processing pipelines
CoGe: Comparative Genomics
Secure | https://genomevolution.org/coge/?ticket=ST-36990-JW9ATK0zRgcjfju5ojUg-auth.iplantc.org
CoGe
Search database
advanced
My Data
Tools
Help

Please take a moment to fill out the **Future Directions Survey** (only 4 questions!) and help guide CoGe's future development: http://goo.gl/nizYYz

Organisms: 17,898 Genomes: 32,823 Features: 1,336,783,172 Experiments: 8,764

New to CoGe?

CoGe is a platform for performing Comparative Genomics research. It provides an open-ended network of interconnected tools to manage, analyze, and visualize next-gen data.

Get started Create an Account Tutorials Documentation FAQ

Tools

OrganismView Search for organisms and get an overview of their genomic make-up. Example - Documentation

EPIC-CoGe Visualize genomes and experiments using a dynamic, interactive genome browser. Example - Documentation

CoGeBlast Blast sequences against any number of organisms in CoGe. Example - Documentation

SynMap Compare any two genomes to identify regions of synteny. Example - Documentation

SynMap3D Compare any three genomes to identify regions of synteny. Example - Documentation

SynFind Search CoGe's annotation database for homologs. Example

GEvo Compare sequences and genomic regions to discover patterns of genome evolution. Example - Documentation

Load Genome Load your own genome from NCBI or a FASTA file. Documentation

Load Experiment (LoadExp+) Load experimental data from various standard input formats (such as BED, WIG, BAM, and FASTQ) and run downstream analyses including read mapping, expression measurement, and SNP identification. Documentation

Recently Added Public Genomes (more)

Tue Sep 26 2017	Scaptomyza flava
Wed Aug 23 2017	hyalella azteca
Wed Aug 09 2017	Plasmodium yoelii 17XNL
Tue Aug 01 2017	Buchnera aphidicola (Chaitophorus capreae)
Mon Jul 17 2017	Drosophila ananassae

Latest News

LoadExp+ July 20th 2017

GC% values added to API July 20th 2017

Multiple fastq support May 9th 2017

Recommend Genomes for Certification April 3rd 2017

NGS Analysis Performance Improvements March 13th 2017

...more...

Worldwide Usage

Tutorials

Questions, problems, suggestions? Contact us or Ask CyVerse

Eric Lyons

Sean Davey

Step 1: Locate, Test & Extend Software

Locating & Testing Appropriate Software

- Locate
  - Peer Review - Search for published reviews/comparisons.
  - Self Review – Test multiple options with own data
- Testing
  - Is this software appropriate for my data?
  - Are the results analytically valid?
  - Are any modifications needed? -> **Extend!**

The CoGe API

A RESTful API for accessing CoGe databases & analytical pipelines

https://genomevolution.org/wiki/index.php/Web_Services_REST_API

- Search, Fetch, Add, Update, Delete…
  - Organisms
  - Genomes
  - Features
  - Experiments
  - Notebooks
  - Groups Jobs
- Obtain JBrowse-compliant data.

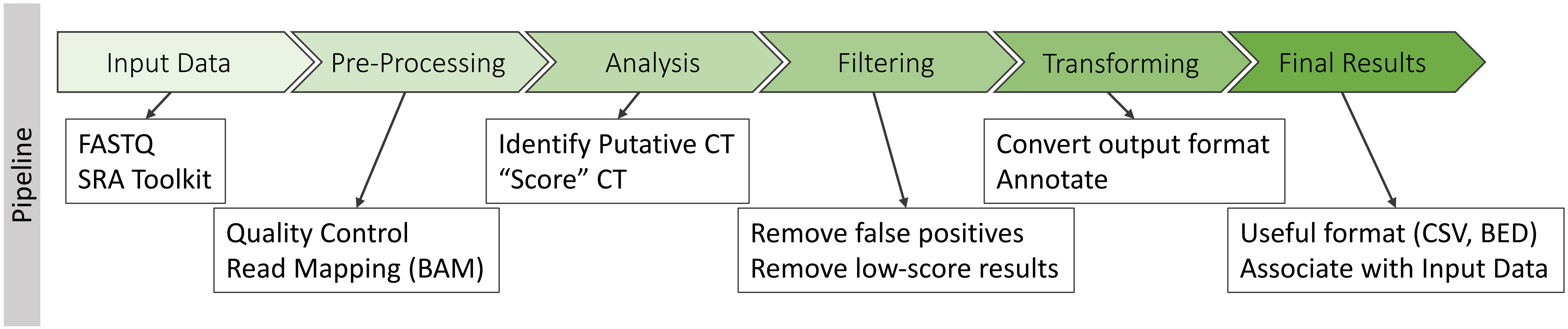
Extending Software

- Not all programs can apply to all organisms.
  - *E.g. Can a human-centric analysis program be modified to work with more complex plant genome data?*
- Not all programs have robust data management.
  - *E.g. If data can’t be stored data, can program be modified to allow remote access?*

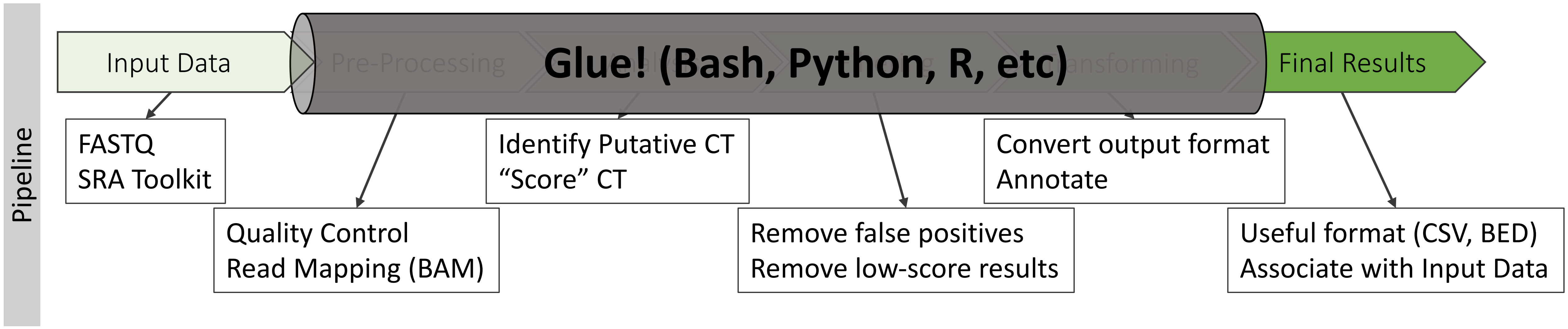
Example: *CoGe Genome Info API Request (Python)*

```
import requests, json

# Base authentication & API URL.
auth = {”user”: “asherkhb”, “token”: XXXXX}
coge = "https://genomevolution.org/coge/api/v1/”

# Request genome information for B. napus (CoGe ID 35008)
gid = 35008
url = coge + “genomes/" + str(gid)
r = requests.get(url, params={'username': user, 'token': token})

# Validate request.
if r.status_code == 200:
    data = json.loads(r.text)
    print(”Success! Obtained data for %s." % data['name']
    chr = data[”chromosomes”]
```

Step 2: Pipeline & Package Analysis Steps

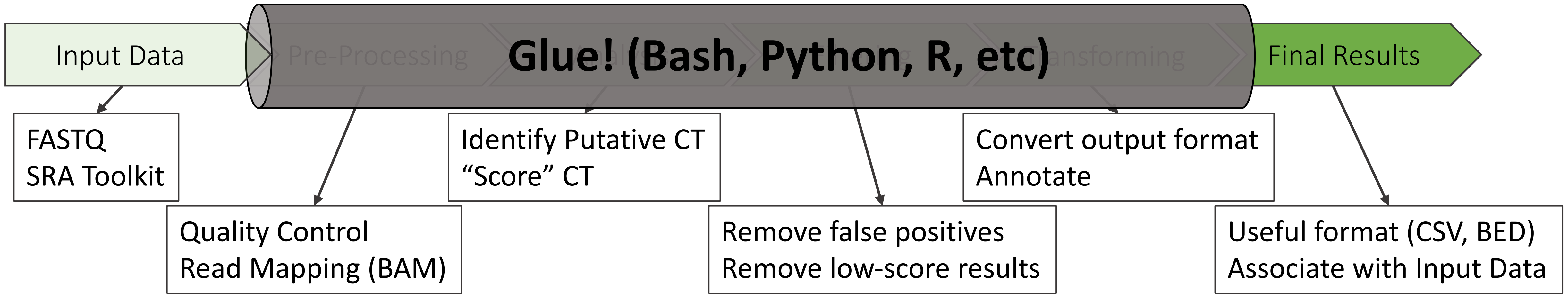


Step 2: Pipeline & Package Analysis Steps



Step 2: Pipeline & Package Analysis Steps

Pipeline



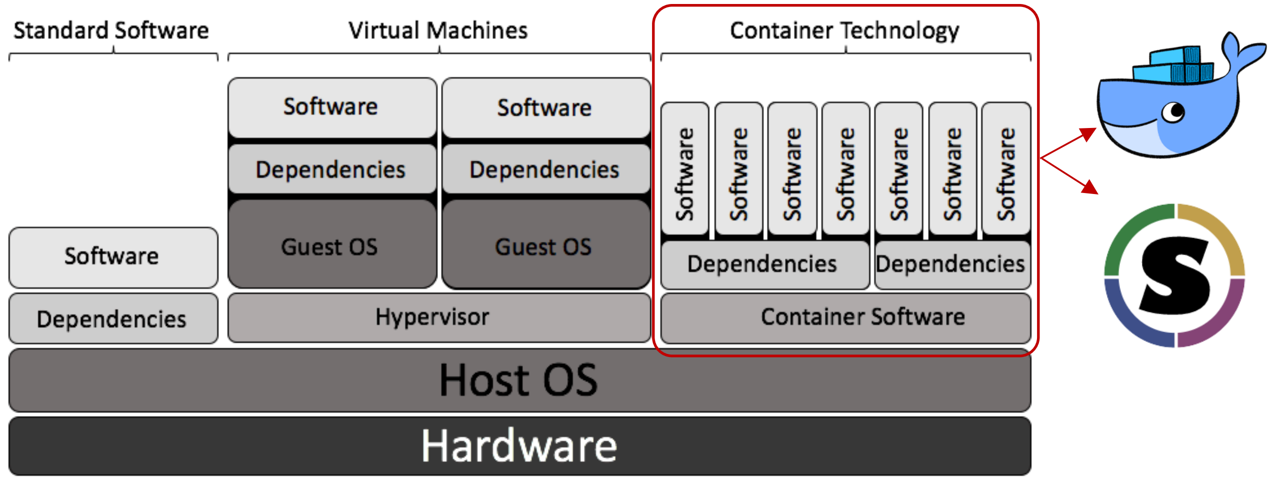
Package

Why "Package"?

- Easier to setup & use.
- Enables scalable deployment.
- Consistent & reproducible.

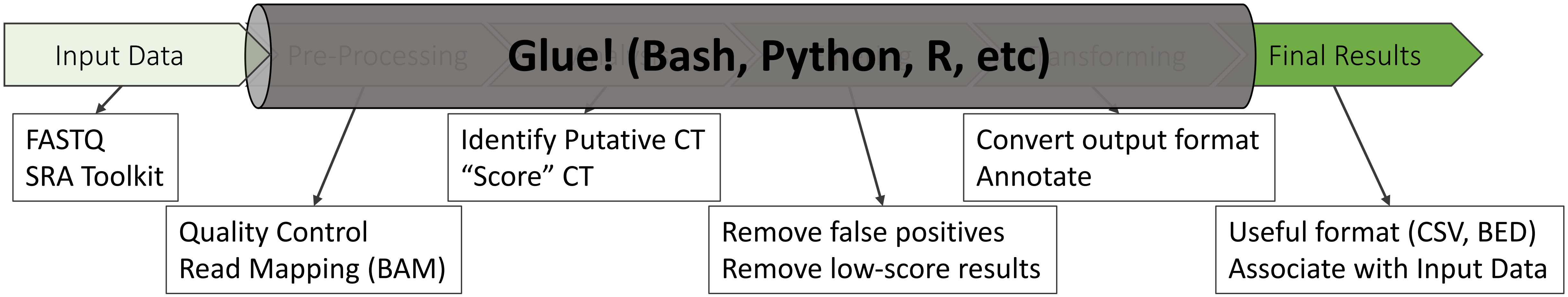
How?

- Containerization
- www.docker.com
- http://singularity.lbl.gov/



Step 2: Pipeline & Package Analysis Steps

Pipeline



Package

Why “Package”?

- Easier to setup & use.
- Enables scalable deployment.
- Consistent & reproducible.

How?

- Containerization
- www.docker.com
- http://singularity.lbl.gov/

docker

```
FROM ubuntu:14.04
RUN apt-get update && apt-get install -y \
    my_dependency_1 my_dependency_2
COPY Code_Repo/ /usr/src/code
WORKDIR /usr/local/data
ENTRYPOINT [”python”, “/usr/src/code/run.py”]
```

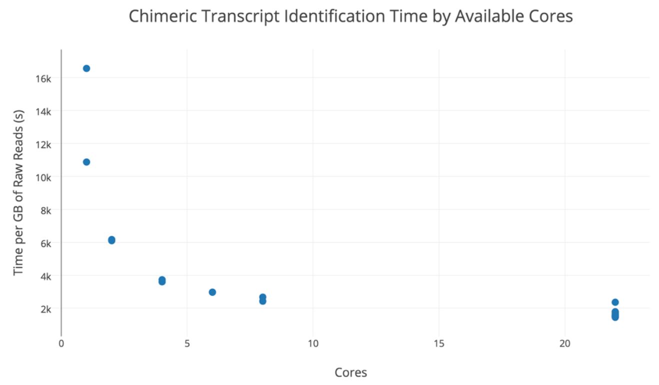
Step 3: Benchmark the Pipeline

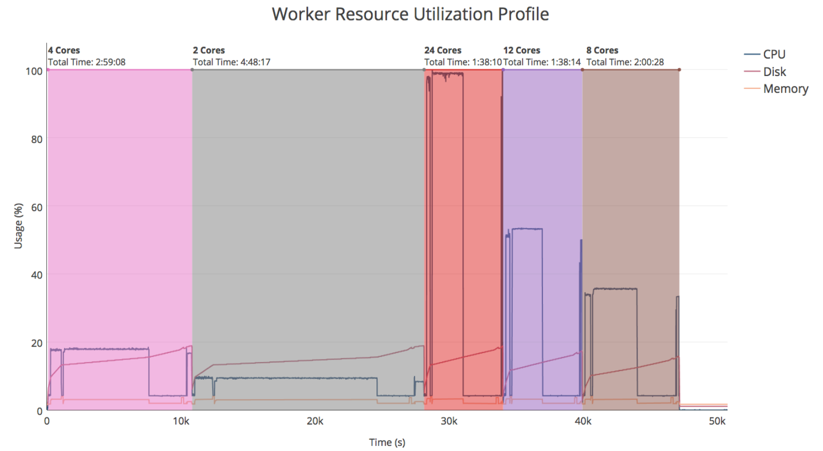
*Benchmarking – testing to **assess relative performance** of a program.*

Memory (RAM) – Storage (Disk) – Time per CPU

<u>Important Benchmarking Questions</u>

1. **How does the program scale, with respects to…**
   - Time & Resources required as input data size increases?
     - *Does doubling input data size require twice the time?*
     - *Does doubling input data require twice the memory/storage capacity?*
   - Time required as resources increase?
     - *Does doubling available cores half the required time?*
2. **How does the program perform over time?**
   - *Are resources used consistently throughout execution, or do different steps have different requirements?*
   - *How do minimum, average, and max(peak) resource requirements differ?*





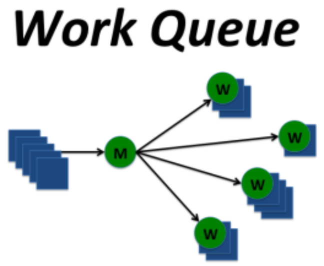
Step 4: Build Coordinating Master Script

Important Considerations

- Ease-of-use
  - Standardized task specification
  - Simple
  - Flexible
- Resource Agnostic/Portable
  - Run on HPC, cloud, laptop, etc.
- Fault-tolerant
  - Retry failed tasks
  - Restart in case of failure

Example: *Simple Configuration*

```
sample_name,refid,sra
At_flower_1,arabidopsis_thaliana,SRR2240277
At_flower_2,arabidopsis_thaliana,SRR2240278
Br_flower_1,brassica_rapa,SRR
```



Step 4: Build Coordinating Master Script

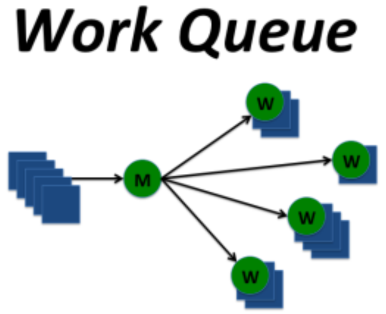
Important Considerations

- Ease-of-use
  - Standardized task specification
  - Simple
  - Flexible
- Resource Agnostic/Portable
  - Run on HPC, cloud, laptop, etc.
- Fault-tolerant
  - Retry failed tasks
  - Restart in case of failure

Example: *Simple Configuration*

```
sample_name,refid,sra
At_flower_1,arabidopsis_thaliana,SRR2240277
At_flower_2,arabidopsis_thaliana,SRR2240278
Br_flower_1,brassica_rapa,SRR
```

"Work Queue is a framework for building large master-worker applications that span thousands of machines drawn from clusters, clouds, and grids."

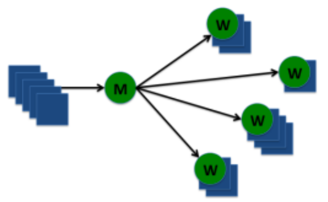


http://ccl.cse.nd.edu/software/workqueue/

CCTools WorkQueue

- Distributed computation framework.
- APIs in Python, Perl, C.
- Simple, but powerful.
- Doesn't require root privilages to install!
- http://ccl.cse.nd.edu/
- https://hub.docker.com/r/asherkhb/workqueue-docker/

Work Queue



Step 4: Build Coordinating Master Script

Important Considerations

- Ease-of-use
  - Standardized task specification
  - Simple
  - Flexible
- Resource Agnostic/Portable
  - Run on HPC, cloud, laptop, etc.
- Fault-tolerant
  - Retry failed tasks
  - Restart in case of failure

CCTools WorkQueue

- Distributed computation framework.
- APIs in Python, Perl, C.
- Simple, but powerful.
- Doesn't require root privilages to install!
- http://ccl.cse.nd.edu/
- https://hub.docker.com/r/asherkhb/workqueue-docker/

Example: *Simple Configuration*

```
sample_name,refid,sra
At_flower_1,arabidopsis_thaliana,SRR2240277
At_flower_2,arabidopsis_thaliana,SRR2240278
Br_flower_1,brassica_rapa,SRR
```

Example: *Basic Python WorkQueue Application*

```
from work_queue import *

tasks = [{"cmd": "head input1.txt > input1.head",
          "inpt": "./input1.txt", "otpt": "input1.head"}, ...]

q = WorkQueue(WORK_QUEUE_DEFAULT_PORT)

for t in tasks:
    t = Task(t[cmd])
    t.specify_file(t[inpt], t[inpt], WORK_QUEUE_INPUT)
    t.specify_file(t[otpt], t[otpt], WORK_QUEUE_OUTPUT)
    q.submit(t)

while not q.empty()
    t = q.wait(5)
```

http://ccl.cse.nd.edu/software/workqueue/

Step 5: Identify & Obtain Compute Resources

Resource Providers

- Local (Personal) Resources (e.g. Laptop)
- Campus Resources: (e.g. HPCs)
- Nationally-funded Resources: (e.g. XSEDE)
- Paid (Private) Resources (e.g. AWS)

Resource Types

- **Local:**
  - Small & limited power, useful for development purposes and testing on small datasets
- **HTC:**
  - Large many core machines, useful for naturally parallizable tasks
- **HTC:**
  - Large many core machines with fast interconnects, useful for jobs requiring taskwise communication
- ⭐ **Cloud:**
  - Scalable, flexible compute resources, useful for jobs requiring variable resources and dynamic scalability.

CYVERSE™ Atmosphere

http://www.cyverse.org/atmosphere

- Free, (nearly) immediate access for researchers.
- Smaller allocations.
- Suitable for development, testing, benchmarking, and projects with small scope.

Jetstream

https://jetstream-cloud.org/

- Free for researchers, must apply for access & allocation.
- Flexible allocation sizes, suitable for projects of all sizes.
- Detailed benchmarking & project justification required to obtain allocation.
- Suitable for large-scale analysis on funded projects.

Step 6: Run Analyses, Collect Results

1. Generate Configuration Files, Run Master
   - Create a small, test configuration file (2-5 tasks).
     - Check connections.
     - Ensure expected behavior.
   - After validation, large scale analysis can begin.
2. Launch Workers
   - Start resources (e.g. VMs)
   - Ensure pipeline is installed & works (you'll appreciate containers here!)
   - Launch workers.

Example: *Launching a Worker*

```
work_queue_worker -d all \               # Print all debug messages
                  --cores 0 \            # Worker can use all resources
                  --workdir ./worker \   # Place files in ./worker
                  123.45.67 9123         # Specify master IP & port
```

https://ccl.cse.nd.edu/software/manuals/man/work_queue_worker.html

3. Monitor Progress
   - You have an automated pipeline, but keep an eye on it!
   - Tracking progress can help…
     - Predict when analyses will be done.
     - Determine number of workers needed.
4. Collect Results
   - WorkQueue handles returning result files, so you can focus on…
   - Making Discoveries!

Step 7: Distribute!

*Distribution is about **ensuring reproducibility** and **enabling further discovery.***

<u>Ensuring Reproducibility</u>

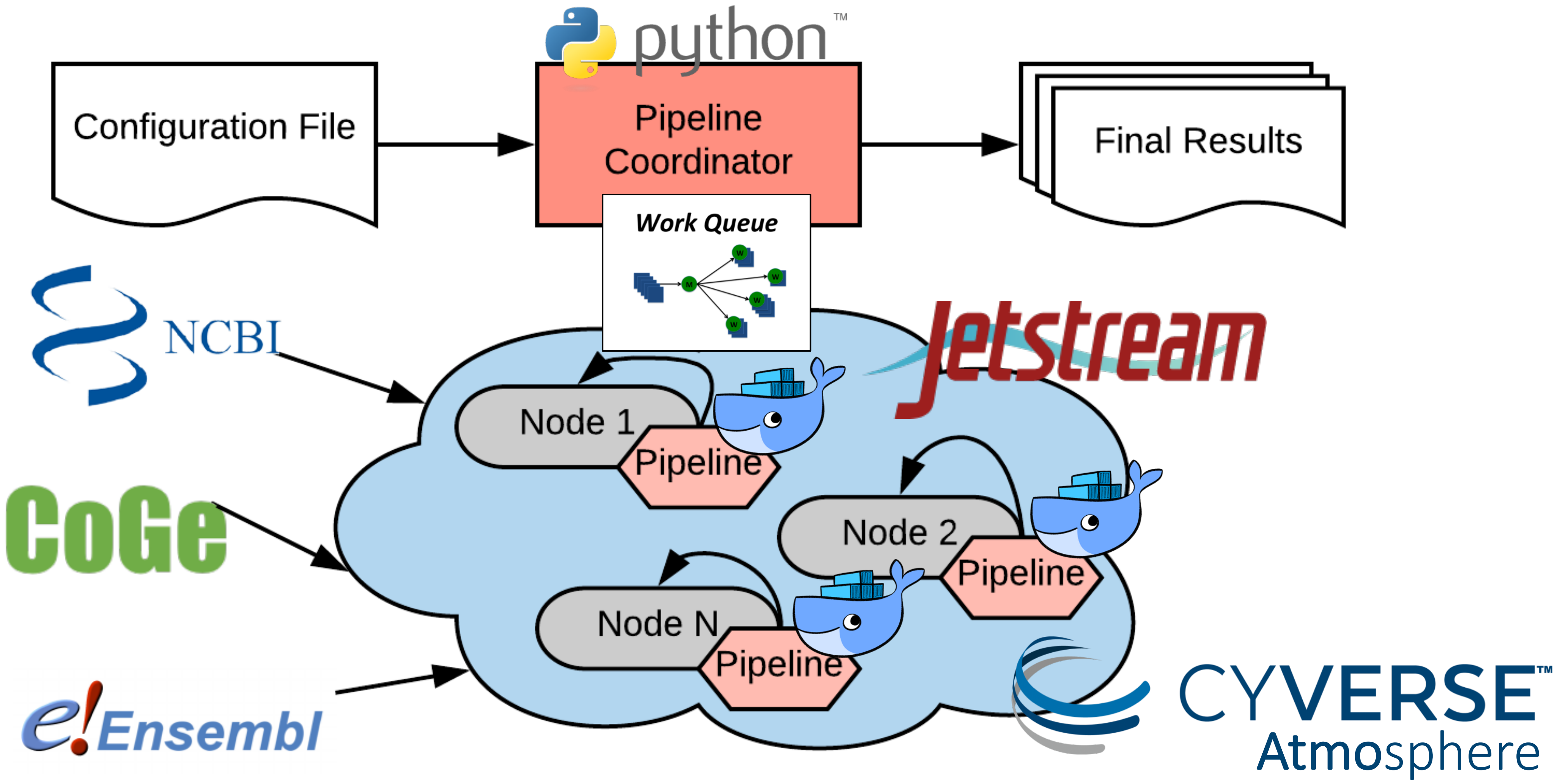
- Open-Source
  - Permissive licenses recommended, but…
  - License may be dictated by institution!
- "Literate Programming"
  - Intersperse detailed comments & descriptive narratives with code to improve understandability.
- Availability
  - Host source code on version control integrated hub.
  - Free & Popular: GitHub, BitBucket

<u>Enabling Further Discovery</u>

1. Documentation
   - Detailed installation instructions.
     - Test instructions on fresh OS (e.g. VM)
     - Include even when packaged dist. available.
   - Clear & concise use instructions.
     - Helpful to have **both** a "quick start" and a detailed use instructions.
   - Example datasets.
   - Tips/Gotchas/FAQs.

2. Accessibility
   - Access to Packaged Version
   - Recommend specific versions
   - Include links to help with packaging technology!
     - E.g. don't assume everyone knows Docker!

Summary


python
Configuration File
Pipeline Coordinator
Final Results
Work Queue
NCBI
Jetstream
Node 1
Pipeline
CoGe
Node 2
Pipeline
Node N
Pipeline
Ensembl
CYVERSE
Atmosphere

Find me Online

www.asherkhb.com

https://github.com/asherkhb/

asherkhb@gmail.com

Important Links

- CoGe API
  - www.genomevolution.org
  - https://genomevolution.org/wiki/index.php/Web_Services_REST_API
- CCTools WorkQueue
  - http://ccl.cse.nd.edu/
  - http://ccl.cse.nd.edu/software/workqueue/
  - https://hub.docker.com/r/asherkhb/workqueue-docker/
- CyVerse Atmosphere
  - http://www.cyverse.org/
  - https://user.cyverse.org/
  - https://atmo.cyverse.org/
- JetStream Cloud
  - https://jetstream-cloud.org/
  - https://portal.xsede.org/submit-request
  - https://use.jetstream-cloud.org/