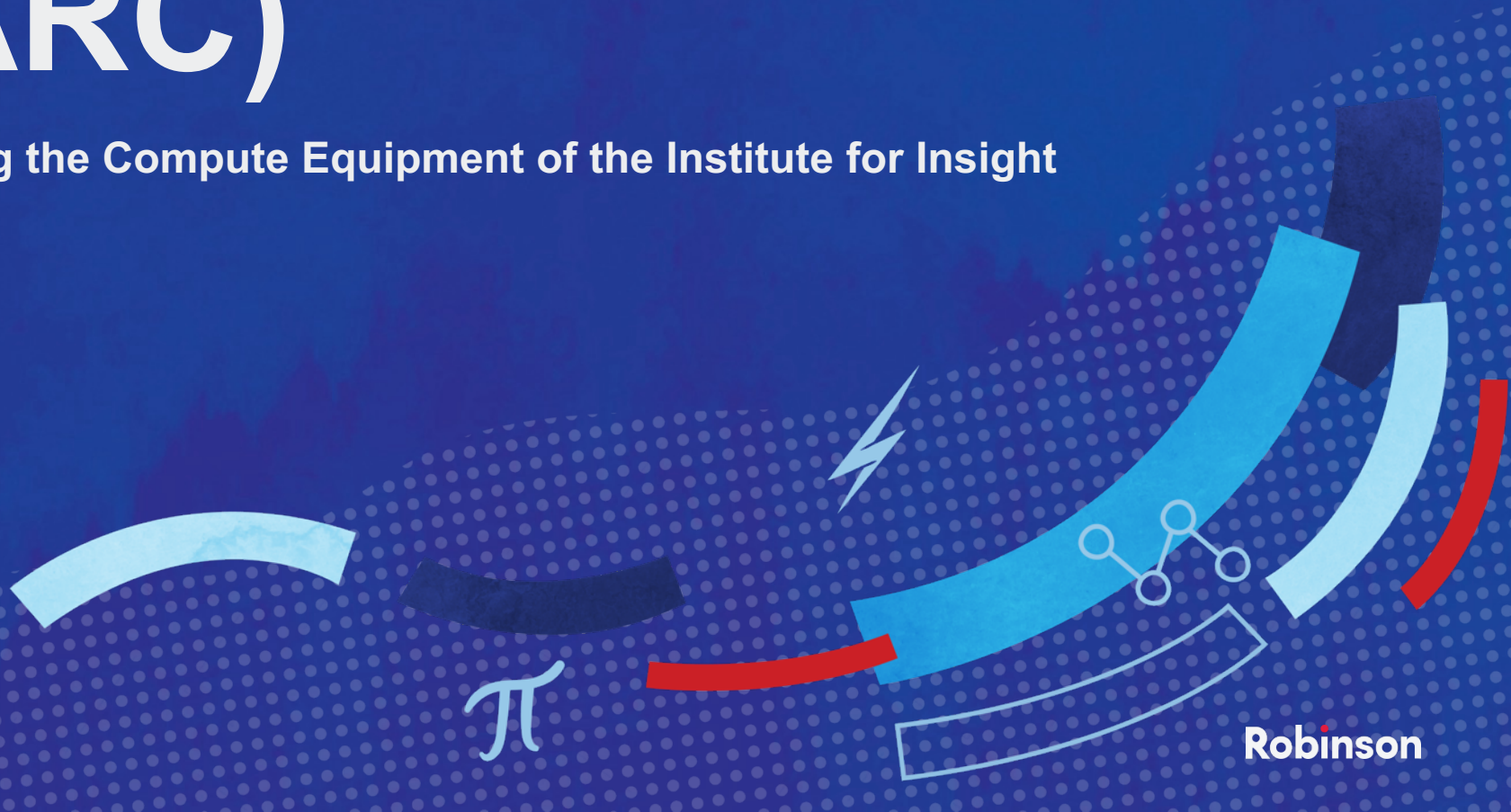
Analytics Research Cluster (ARC)

Overview and Introduction to Using the Compute Equipment of the Institute for Insight

Péter Molnár pmolnar@gsu.edu

October 21, 2023

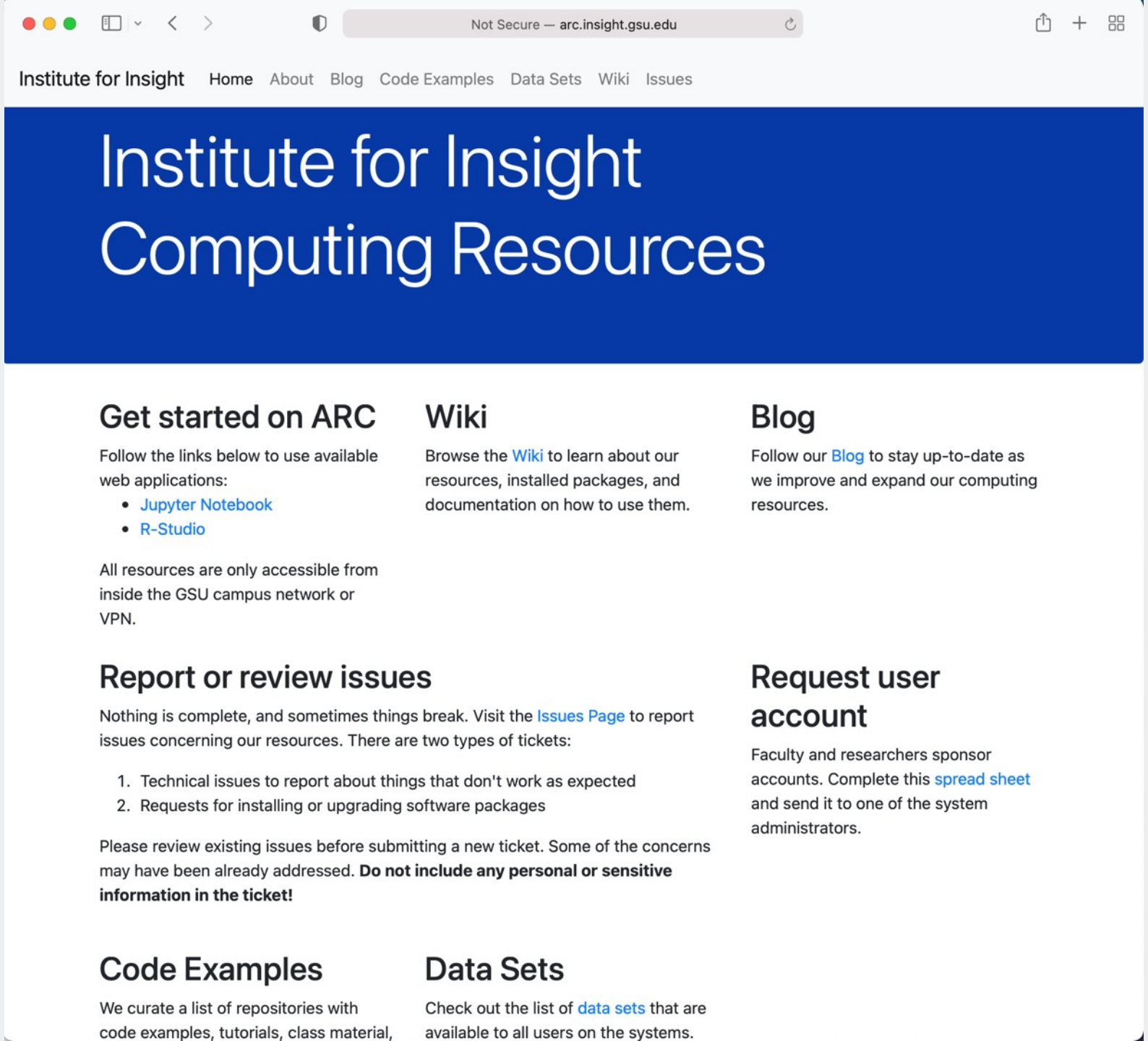


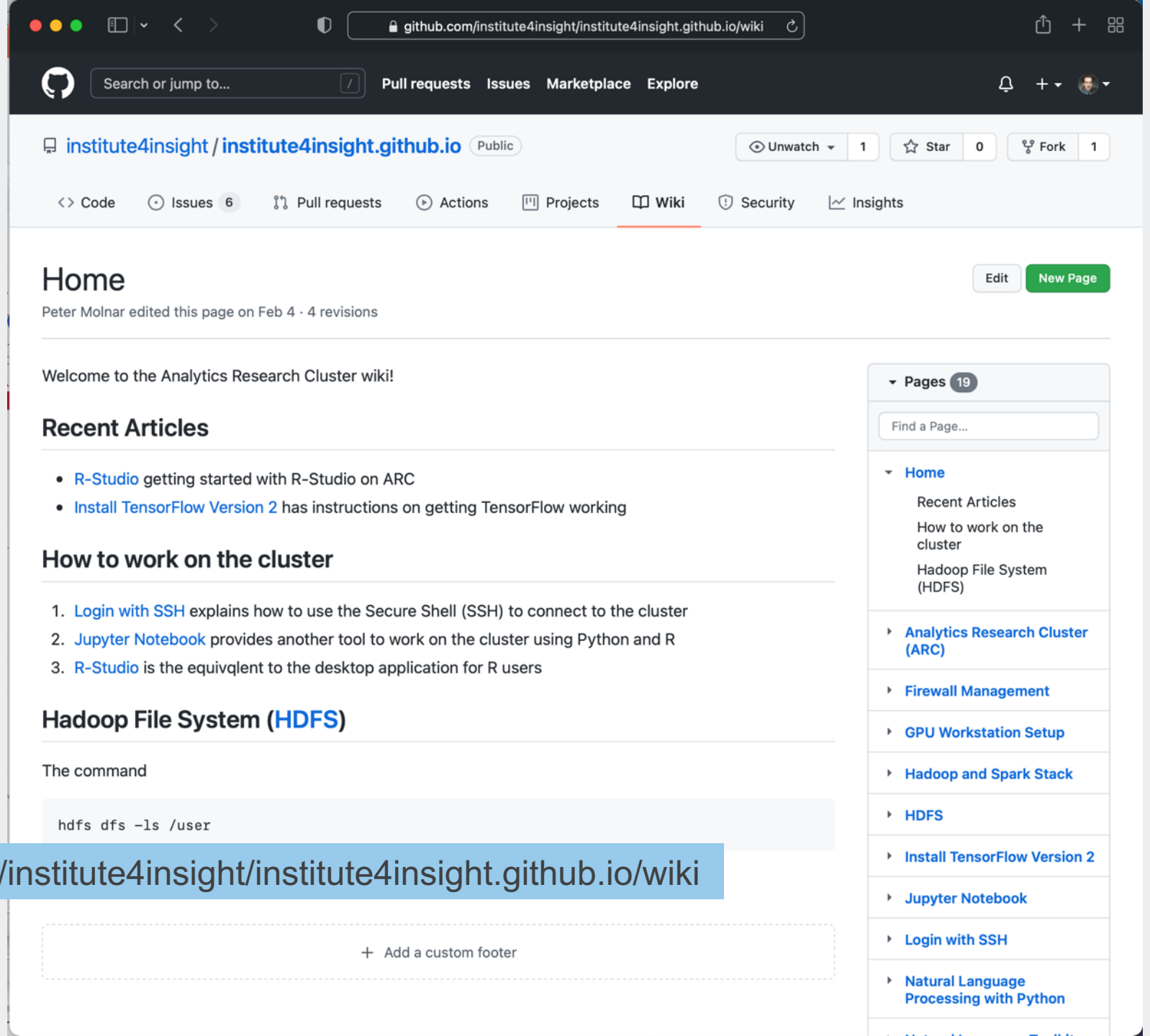
Georgia State University | J. Mack Robinson College of Business

Robinson

Outline

Section
Cluster Architecture
UNIX Command Line Interface (CLI)
Remote Access via Secure Shell and Secure File Transfer
Introduction to Jupyter Lab
File Systems
SQL Database Servers
Overview of Distributed Computing


Not Secure — arc.insight.gsu.edu
Institute for Insight
Home
About
Blog
Code Examples
Data Sets
Wiki
Issues
Institute for Insight
Computing Resources
Get started on ARC
Follow the links below to use available web applications:
Jupyter Notebook
R-Studio
All resources are only accessible from inside the GSU campus network or VPN.
Wiki
Browse the Wiki to learn about our resources, installed packages, and documentation on how to use them.
Blog
Follow our Blog to stay up-to-date as we improve and expand our computing resources.
Report or review issues
Nothing is complete, and sometimes things break. Visit the Issues Page to report issues concerning our resources. There are two types of tickets:
1. Technical issues to report about things that don't work as expected
2. Requests for installing or upgrading software packages
Please review existing issues before submitting a new ticket. Some of the concerns may have been already addressed. Do not include any personal or sensitive information in the ticket!
Request user account
Faculty and researchers sponsor accounts. Complete this spread sheet and send it to one of the system administrators.
Code Examples
We curate a list of repositories with code examples, tutorials, class material,
Data Sets
Check out the list of data sets that are available to all users on the systems.



https://github.com/institute4insight/institute4insight.github.io/wiki

Cluster Architecture

Introduction to the Analytics Research Cluster (ARC)

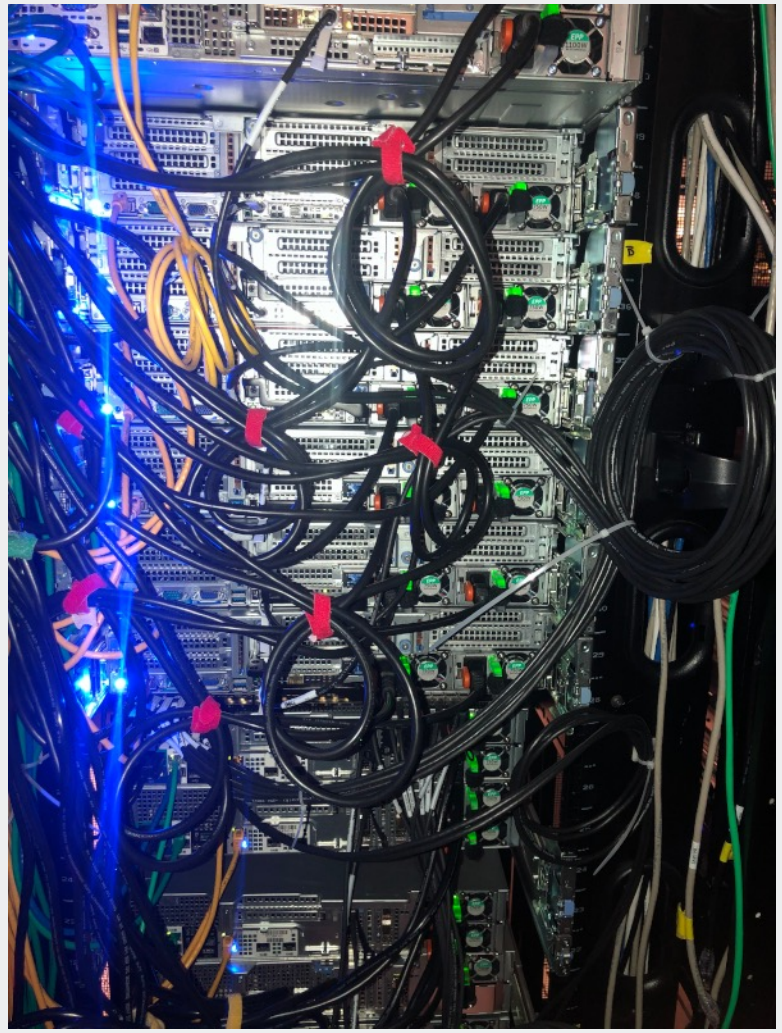
October 21, 2023

Georgia State University | J. Mack Robinson College of Business

Robinson

Analytics Research Cluster

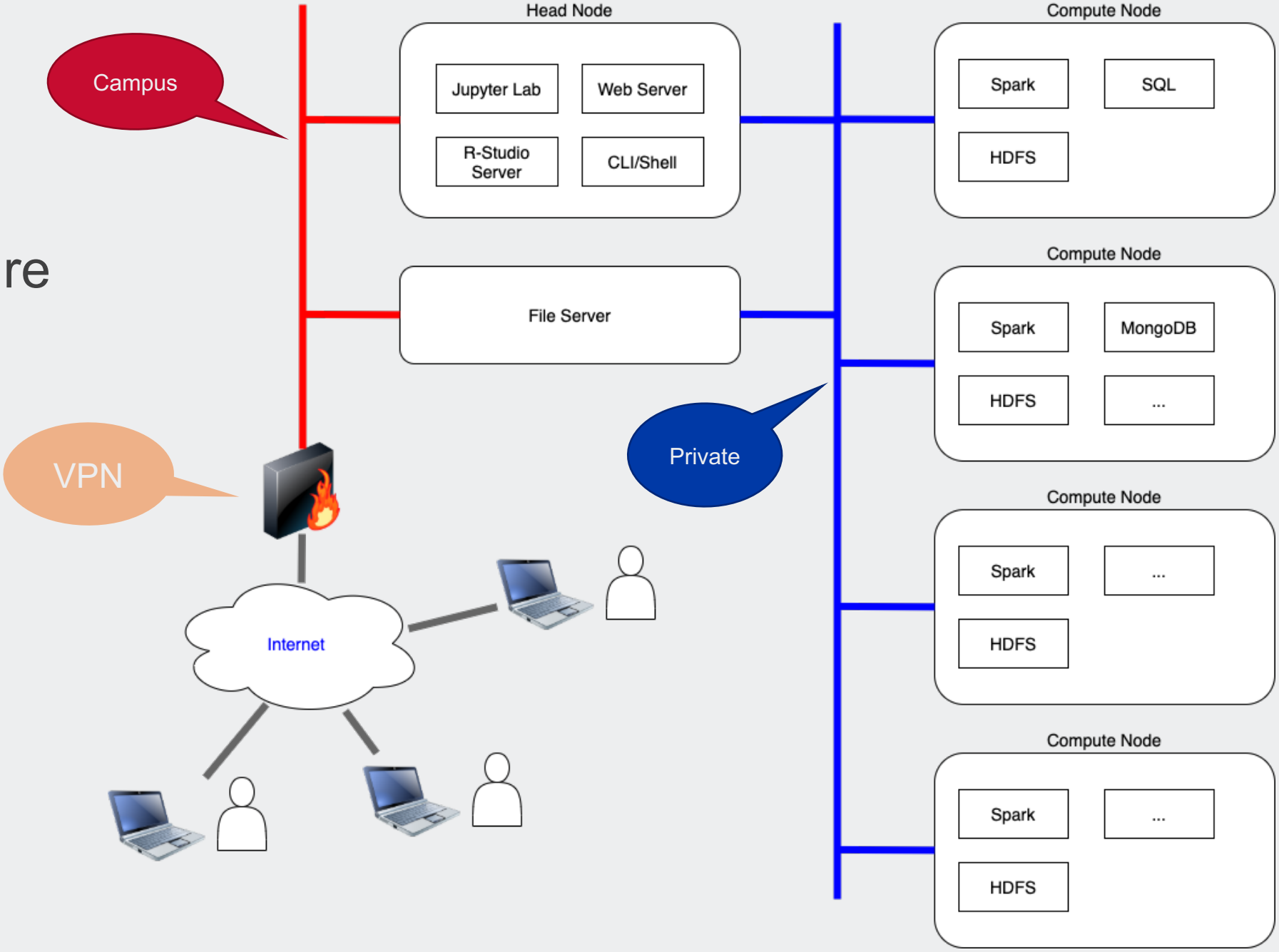
in the
GSU Data Center



Georgia State University | J. Mack Robinson College of Business

Robinson

Network Architecture



UNIX Command Line Interface

Introduction to the Analytics Research Cluster (ARC)

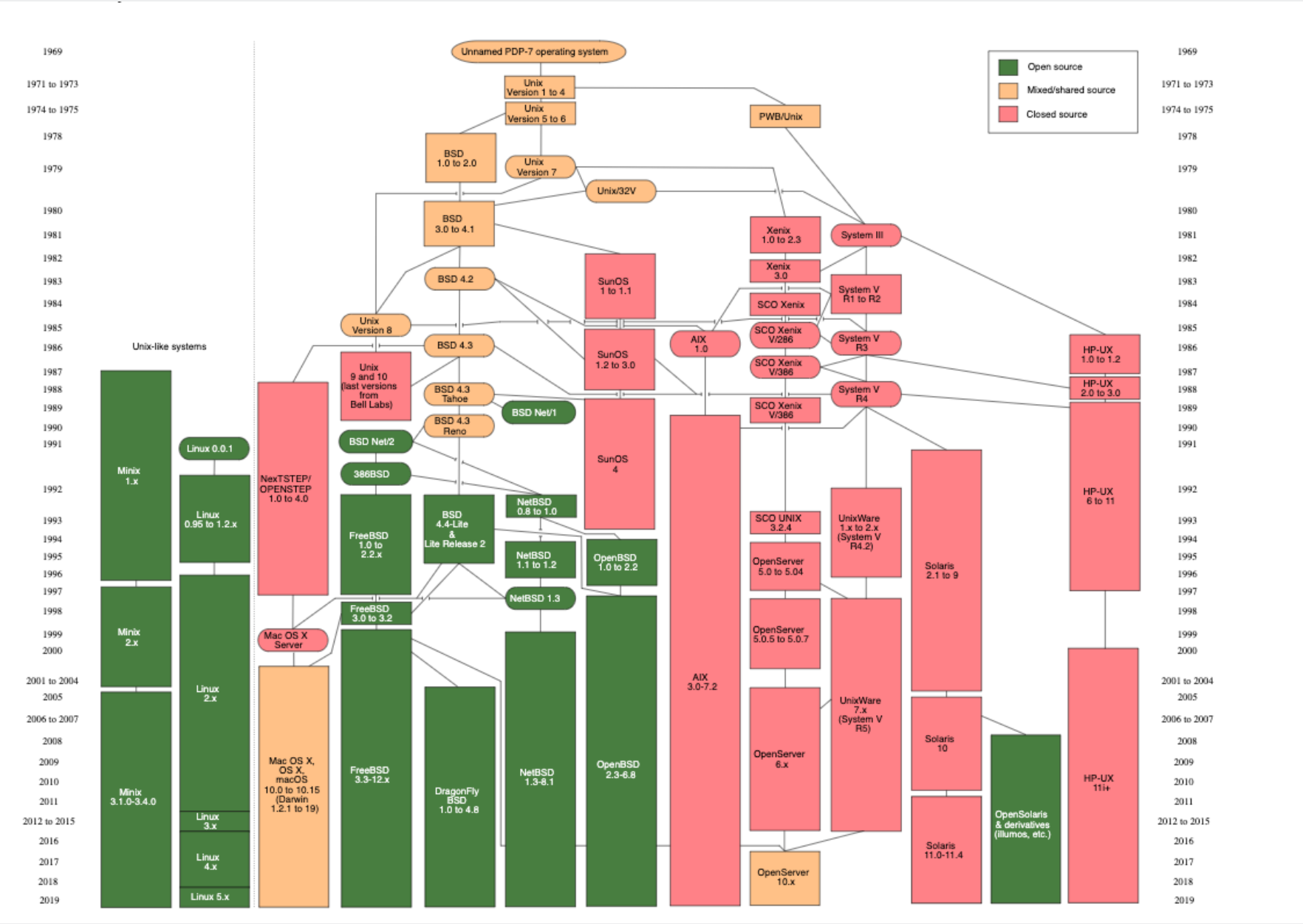
October 21, 2023

Georgia State University | J. Mack Robinson College of Business

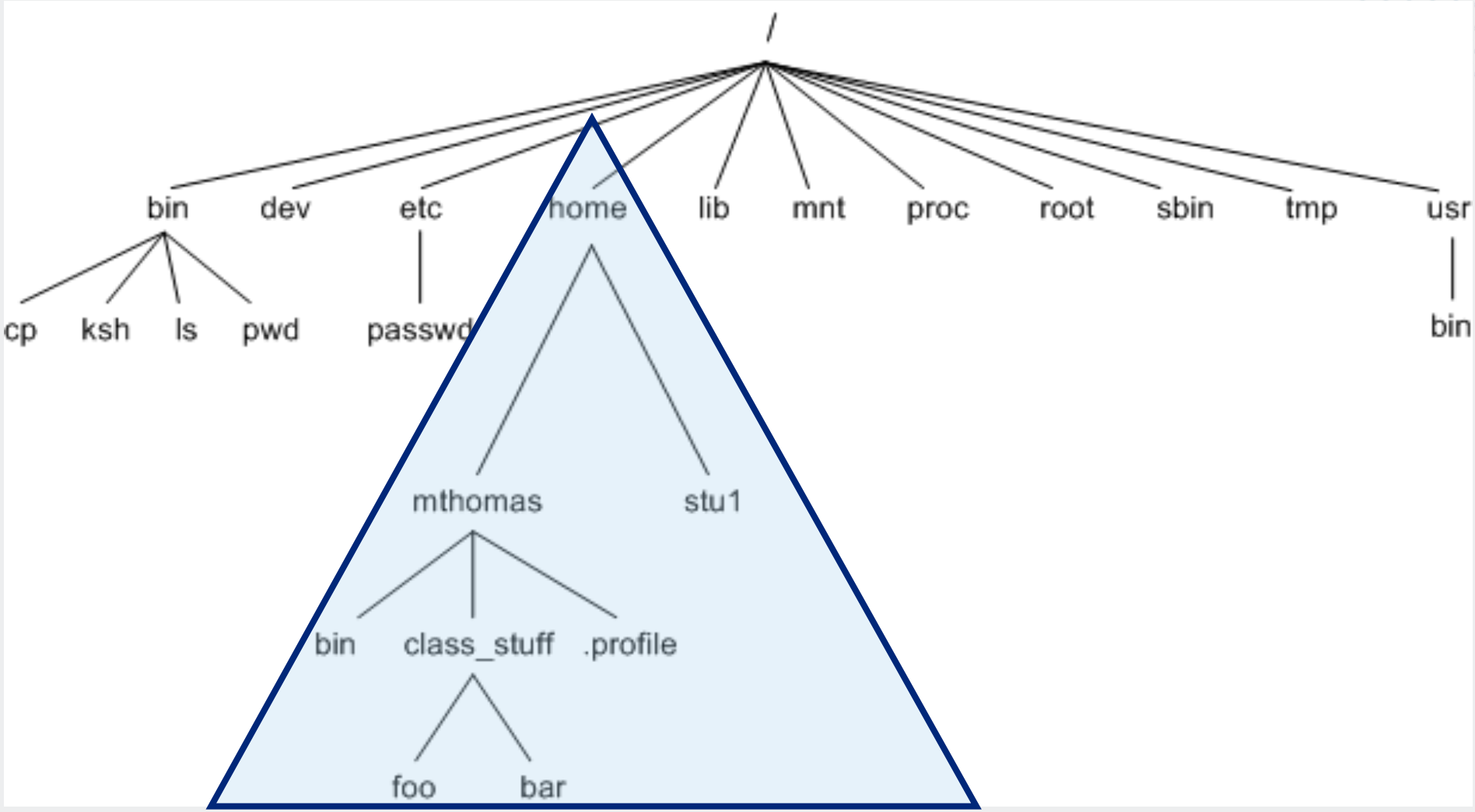
Robinson

Overview and Demo on GitHub

https://github.com/kingmolnar/DataScienceProgramming/blob/master/01-Intro-UNIX/Introduction_orig.ipynb


Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
SunOS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
AIX 1.0
System V R3
BSD 4.3
SunOS 1.2 to 3.0
SCO Xenix V/386
HP-UX 1.0 to 1.2
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
BSD Net/1
System V R4
HP-UX 2.0 to 3.0
SCO Xenix V/386
BSD 4.3 Reno
BSD Net/2
Linux 0.0.1
SunOS 4
386BSD
NetBSD 0.8 to 1.0
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
HP-UX 6 to 11
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 9
NetBSD 1.3
FreeBSD 3.0 to 3.2
OpenServer 5.0.5 to 5.0.7
UnixWare 7.x (System V R5)
Solaris 10
OpenServer 6.x
OpenSolaris & derivatives (illumos, etc.)
Solaris 11.0-11.4
OpenServer 10.x
HP-UX 11i+
Open source
Mixed/shared source
Closed source
Unix-like systems
Minix 1.x
Minix 2.x
Minix 3.1.0-3.4.0
Linux 0.95 to 1.2.x
Linux 2.x
Linux 3.x
Linux 4.x
Linux 5.x
NextSTEP/ OPENSTEP 1.0 to 4.0
Mac OS X Server
Mac OS X, OS X, macOS 10.0 to 10.15 (Darwin 1.2.1 to 19)
FreeBSD 1.0 to 2.2.x
FreeBSD 3.3-12.x
BSD 4.4-Lite & Lite Release 2
DragonFly BSD 1.0 to 4.8
NetBSD 1.3-8.1
OpenBSD 2.3-6.8
AIX 3.0-7.2
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2015
2016
2017
2018
2019

UNIX Directory Tree



Remote Access with SSH and SFTP

Introduction to the Analytics Research Cluster (ARC)

October 21, 2023

Georgia State University | J. Mack Robinson College of Business

Robinson

Login with SSH

SSH & SFTP The server can be accessed via the Secure Shell protocol for command line use and file transfer.

Popular clients are:

1. The ssh and scp commands on POSIX systems
2. File transfer tools like FileZilla (OSX, Windows, Linux) and WinSCP (Windows)
3. Terminal program PuTTY (Windows)...though, the Jupyter Notebook also provides a terminal Users may add public keys to their .ssh/authorized_keys file. Do not remove the existing keys that are needed to utilize the client nodes.

Read how to setup password-less SSH.
https://github.com/institute4insight/institute4insight.github.io/wiki/Login-with-SSH

How to setup password-less SSH

Configuring password-less SSH is crucial for working across multiple (UNIX) systems.

password-less means that SSH authenticates a connection by using private/public key-pairs from the respective hosts, and won't prompt for a password.

Using the CLI users can easy start programs on remote machines with commands like

```
$ ssh remote_host run_analysis.py or copy files back and forth between systems

$ scp -r my_data remote_host:/local/data/foo Creating key pairs on Mac OS and Linux
```

All users need to do is open the Terminal window and type the command

```
$        ssh-keygen
```

The program prompts for input, hit to accept the defaults (which will be just fine).
For detailed https://www.macworld.co.uk/how-to/mac-software/how-generate-ssh-keys-3521606/

How to setup password-less SSH (Windows)

Creating key pairs on MS Windows

The terminal software SmartTTY http://smartty.sysprogs.com will perform the setup automatically when the "password-less" option is checked.

Alternatively, there is a set of PuTTY commands that can be downloaded from https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Instructions can be found at https://docs.joyent.com/public-cloud/getting-started/ssh-keys/generating-an-ssh-key-manually/manually-generating-your-ssh-key-in-windows

Configure system to allow password-less login

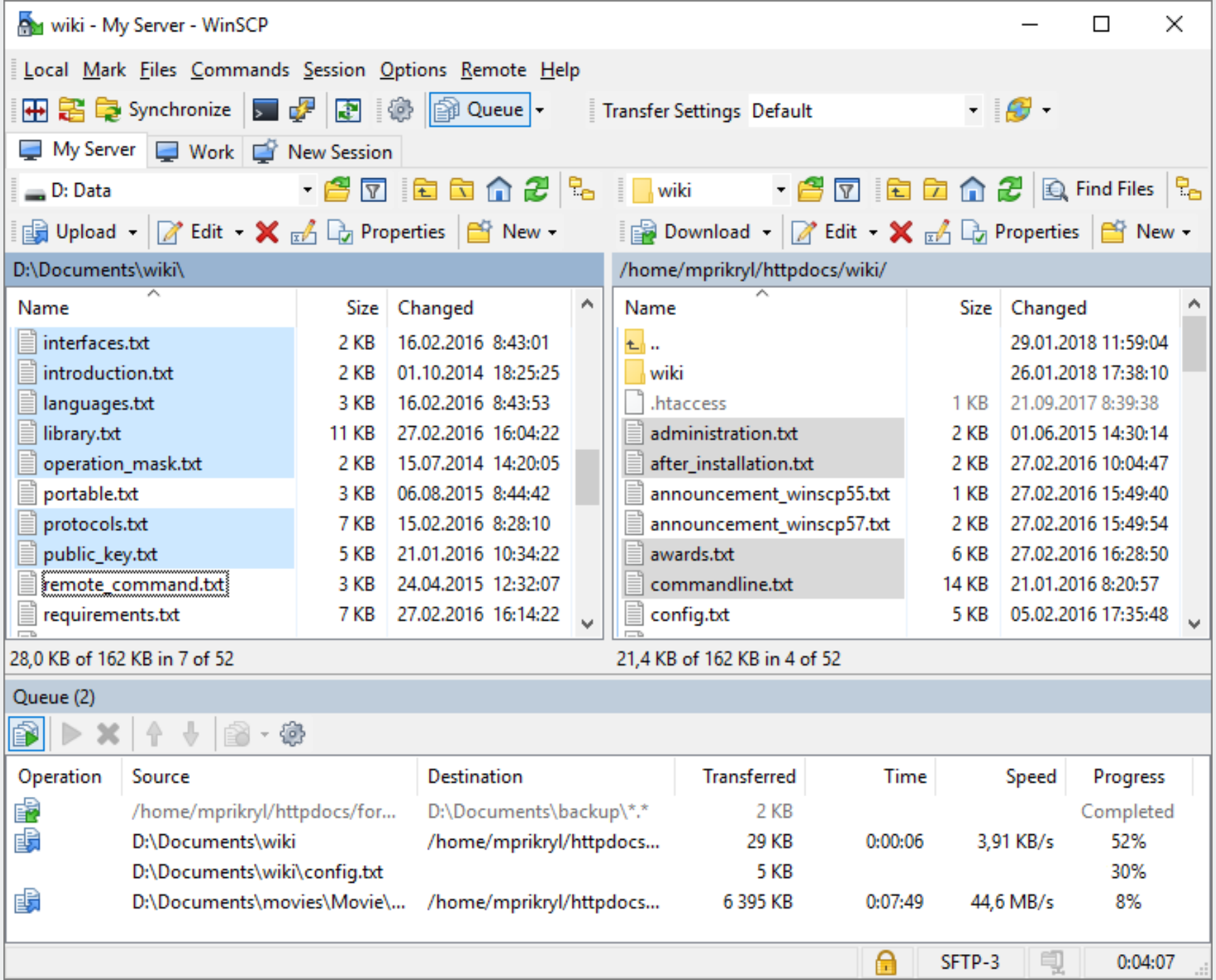
This following outlines the steps to configure password-less login.

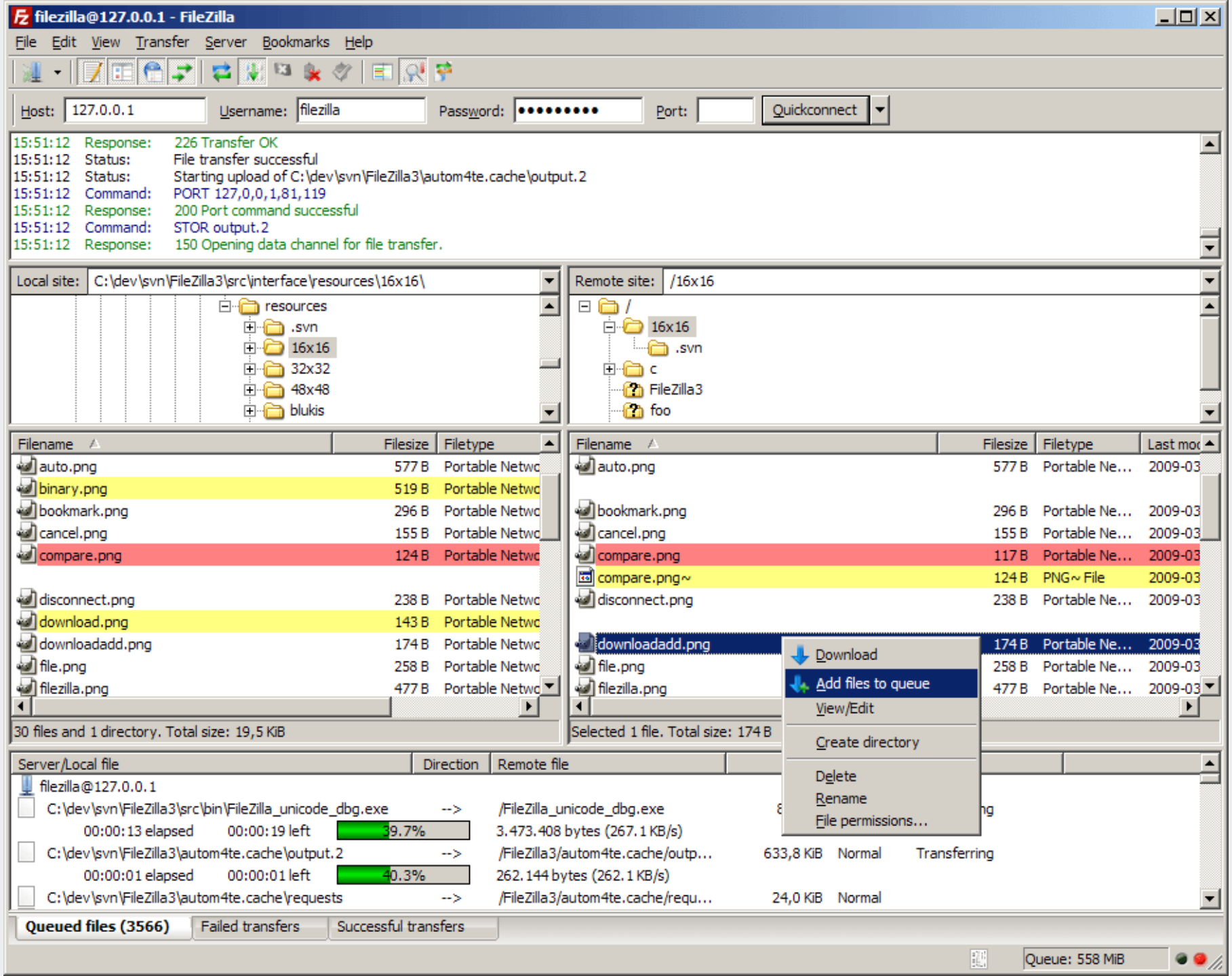
1. create a private/public key-pair on the user's workstation (from which to connect to a *remote* server)
2. copy the public key (by default the file .ssh/id_rsa.pub to the remote system
3. login to the remote system, and
    1. create a `.ssh` directory on the user's home directory, if it doesn't exist
    2. add the public key from the workstation to the end of the file `~/.ssh/authorized_keys`
    (Make sure not to overwrite any existing entries!)

Secure File Transfer Protocol (SFTP)

Some popular SFTP clients

- https://filezilla-project.org (Windows, MacOS, Linux)
- https://winscp.net/eng/index.php (Windows)
- https://mobaxterm.mobatek.net (Windows)





Georgia State University | J. Mack Robinson College of Business

Robinson

SSH Tunnel

SSH tunnels can be used when services like SQL-Server or the Spark UI cannot be accessed directly due to firewall rules.

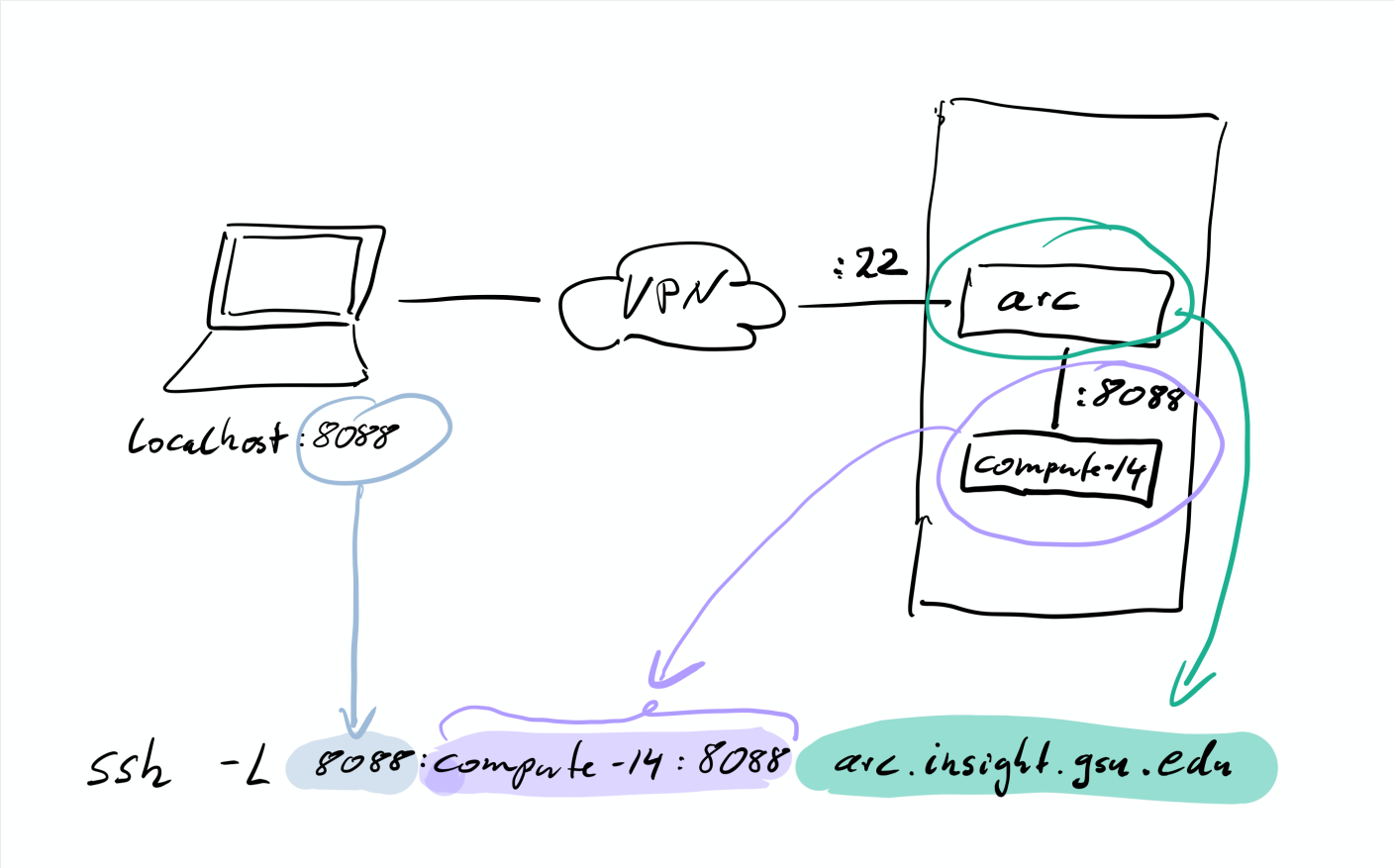
For example: accessing the Spark History Server on compute-14.insight.gsu.edu that runs on port 18080 requires a tunnel.

Some systems like Hadoop and Spark provide convenient web-interfaces. However, these interfaces pose a security risk if they were exposed to the Internet.

The use of SSH tunnels ensures that only authorized users have access. We protect SQL servers in the same way.

https://github.com/institute4insight/institute4insight.github.io/wiki/SSH-Tunnel

SSH Tunnel (2)


VPN
:22
arc
:8088
compute-14
localhost:8088
ssh -L 8088:compute-14:8088 arc.insight.gsu.edu

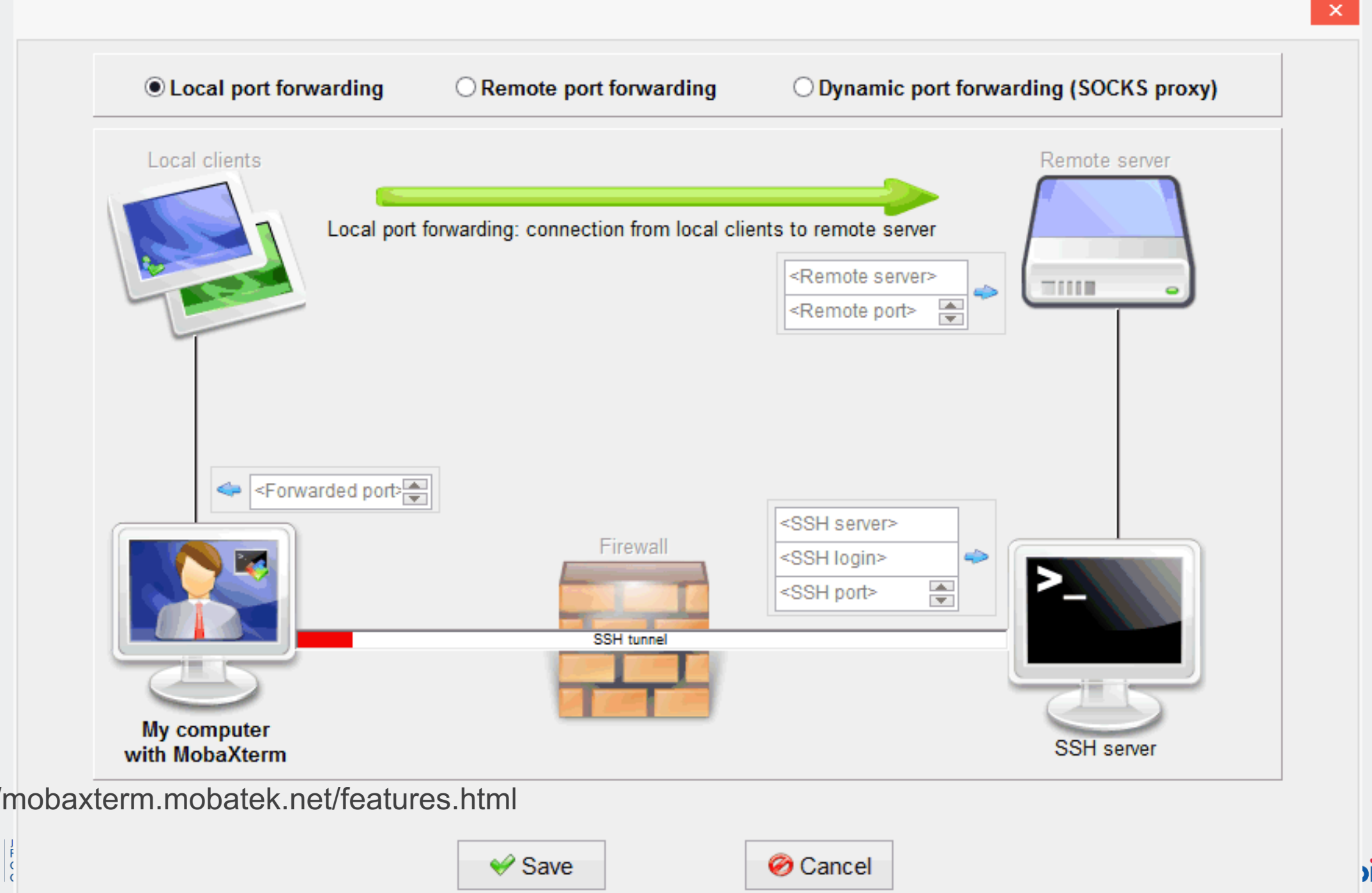
SSH Tunnel (3)

Useful SSH Tunnels

Yarn Manager and Spark UI Proxy	ssh -fNL 8088:compute-14:8088 arc.insight.gsu.edu
Spark History Server	ssh -fNL 18080:compute-14:18080 arc.insight.gsu.edu
PostgreSQL Server	ssh -fNL 5432:sqlserver-1:5432 arc.insight.gsu.edu

Utilities for SSH Tunnels

- CLI command ssh is available on Apple Mac OS, Linux and any other UNIX variation
- Windows Subsystem for Linux https://docs.microsoft.com/en-us/windows/wsl/about

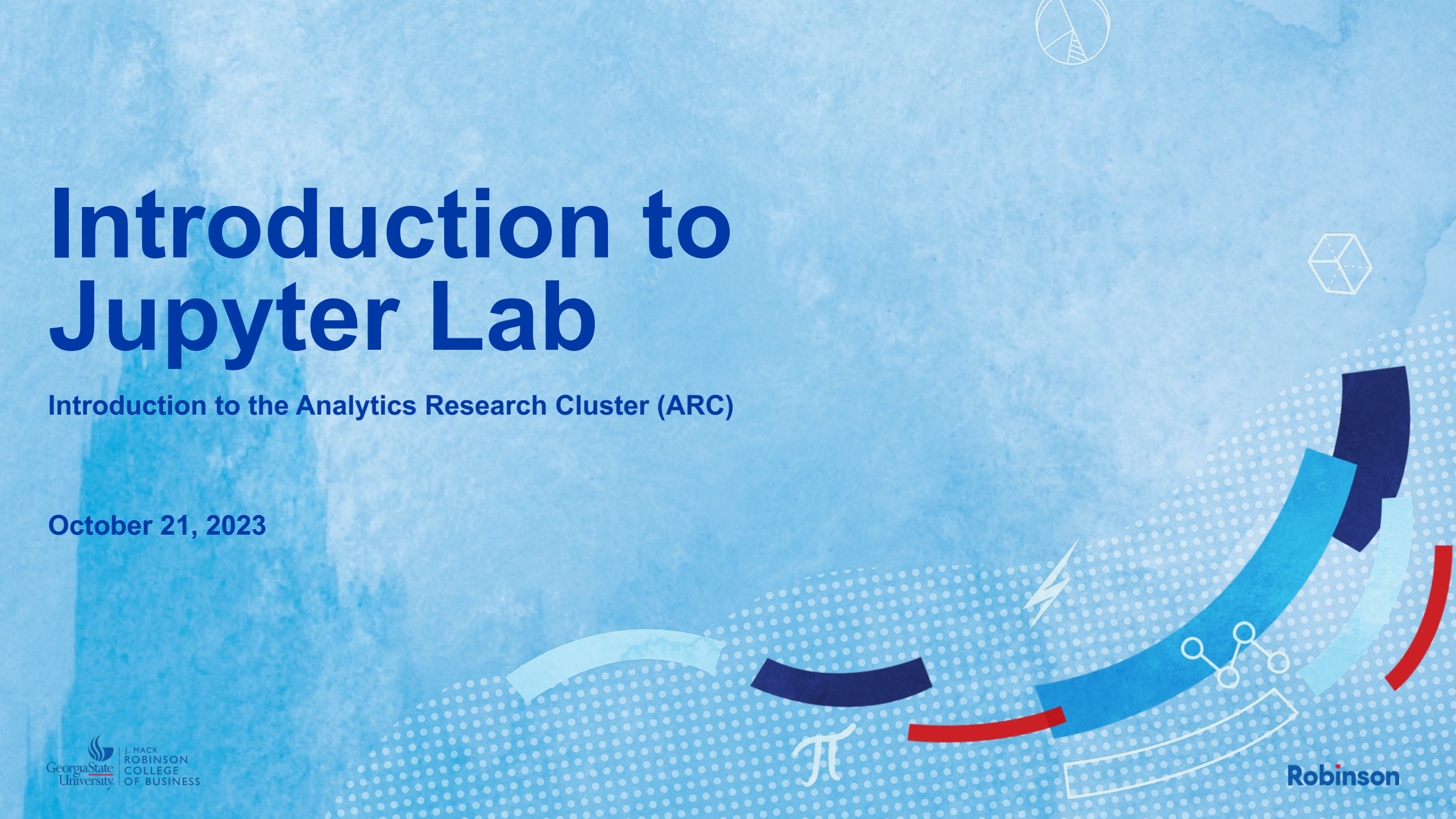


https://mobaxterm.mobatek.net/features.html

Introduction to Jupyter Lab

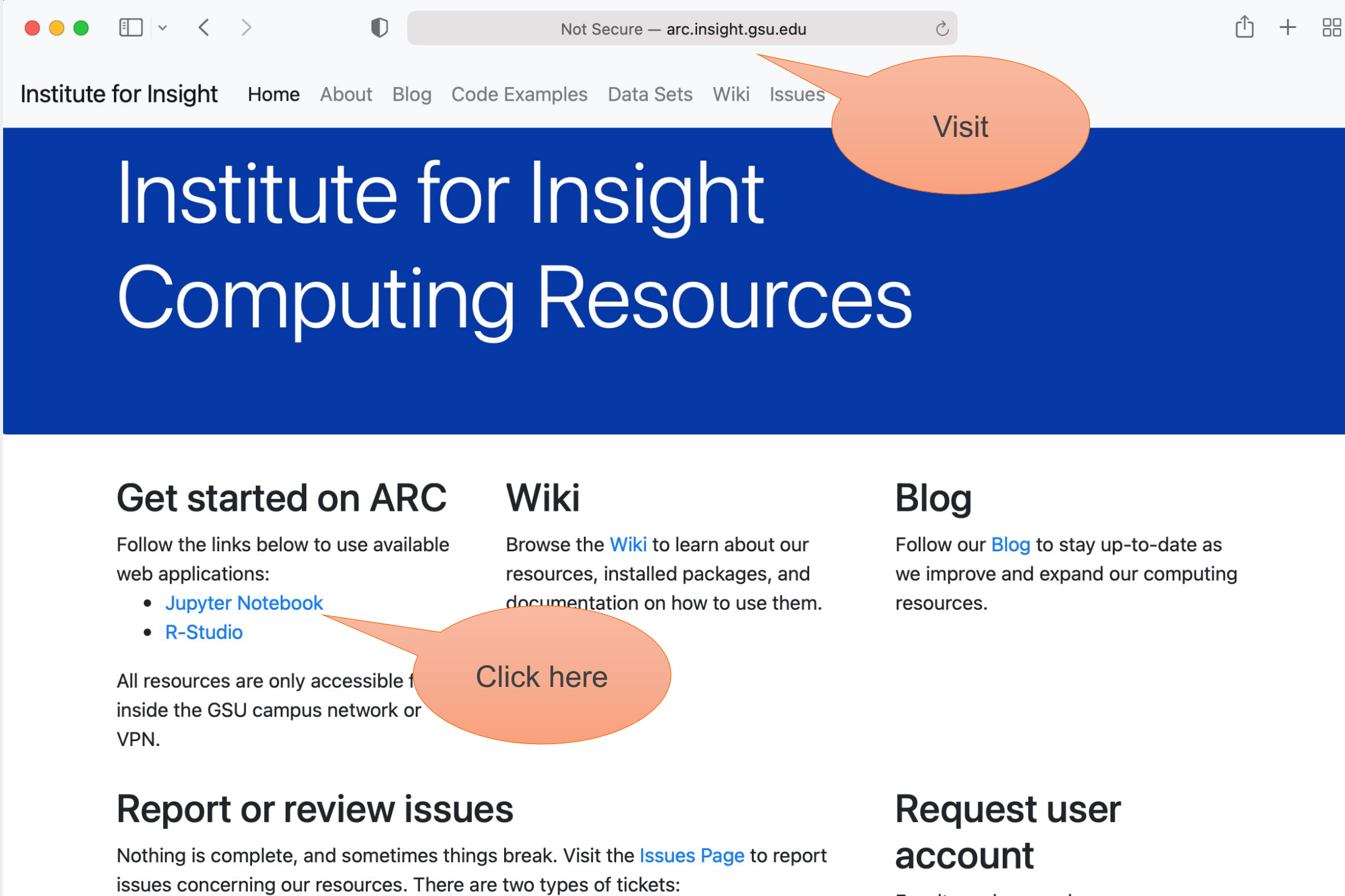
Introduction to the Analytics Research Cluster (ARC)

October 21, 2023



Georgia State University | J. Mack Robinson College of Business

Robinson



Georgia State University

son

File Systems

Introduction to the Analytics Research Cluster (ARC)

October 21, 2023

Georgia State University | J. Mack Robinson College of Business

Robinson

File Systems

Local File System

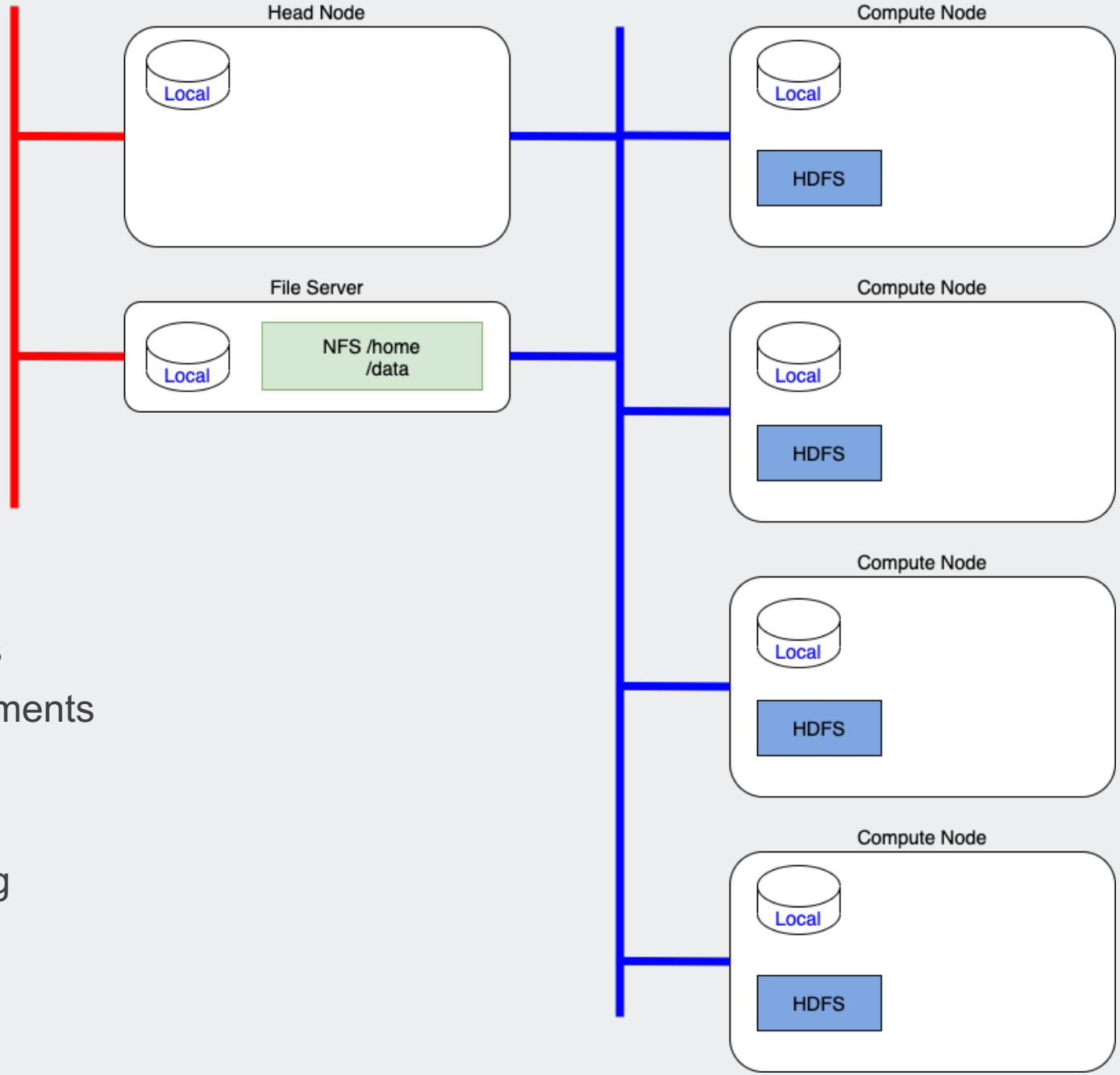
- Every node has their own
- Operating systems, software

Network File System

- Shared across multiple nodes
- `/data` archive of large data sets
- `/home` user programs and documents

Hadoop File System

- Distributed for parallel processing



SQL Databases

Introduction to the Analytics Research Cluster (ARC)

October 21, 2023

Georgia State University | J. Mack Robinson College of Business

Robinson

Outline

Structured Query Language (SQL)

Server Architecture

User Authentication

Desktop Applications

Query via Python

Limitations


SQL

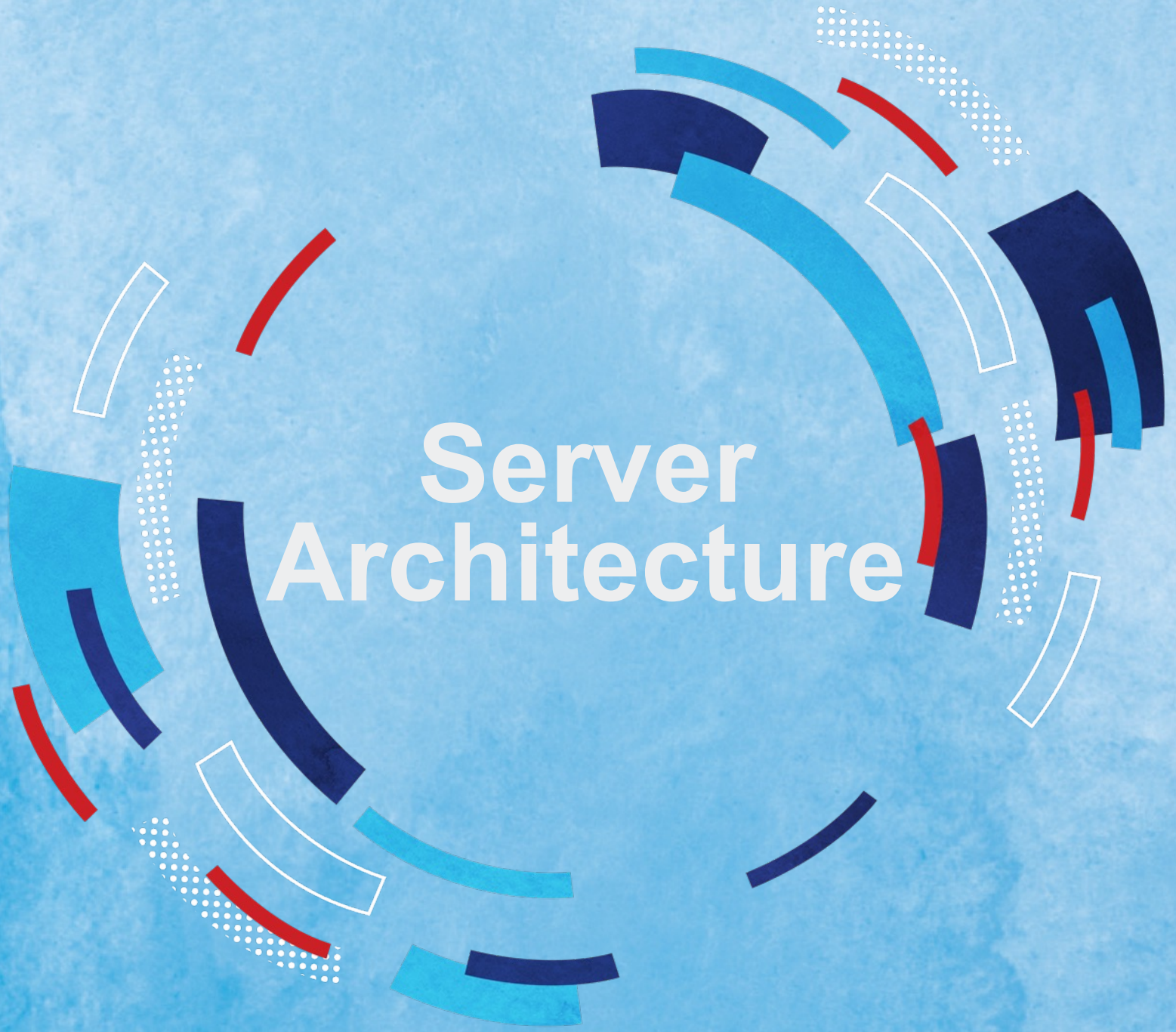
SQL – Select Statement

```
5) SELECT CONCAT(country, site) AS location,
       DATEPART(YEAR, sales_date) AS yr,
       SUM(sales_qty) AS total_sales
1) FROM company_sales
2) WHERE sales_status = 'completed'
3) GROUP BY CONCAT(country, site), DATEPART(YEAR, sales_date)
4) HAVING COUNT(total_sales) >= 10
6) ORDER BY total_sales
7) LIMIT 100
```

ORDER	CLAUSE	FUNCTION
1	from	Choose and join tables to get base data.
2	where	Filters the base data.
3	group by	Aggregates the base data.
4	having	Filters the aggregated data.
5	select	Returns the final data.
6	order by	Sorts the final data.
7	limit	Limits the returned data to a row count.

Assign as much workload as possible to the SQL server…

Server Architecture



Authentication

Connection and Authentication

(Traditionally) you connect to SQL Servers via network connection

Never include your database login and password in your program code.
Keep them in a separate file or secret manager.
Never commit credentials file to Git repo!

```
# load database credentials and change database to "datalabeling"
db_creds = json.load(open(f"/home/{os.environ['USER']}/.arc_config/databases/sqlserver-2-3306.txt"))
connection_uri = '/'.join( db_creds['default_uri'].split('/')[:3] + ['datalabeling'])
```

Desktop Apps

Desktop SQL Apps

Applications to query and manage SQL databases

Some support many different relational database systems – additional connectors (drivers) may be required

Others only support a specific database system: MySQL Workbench, MS SQL-Server Management Studio

Useful for SQL (only) querying and analysis of data. Explore database tables. Create and test SQL queries that are used (Python) code

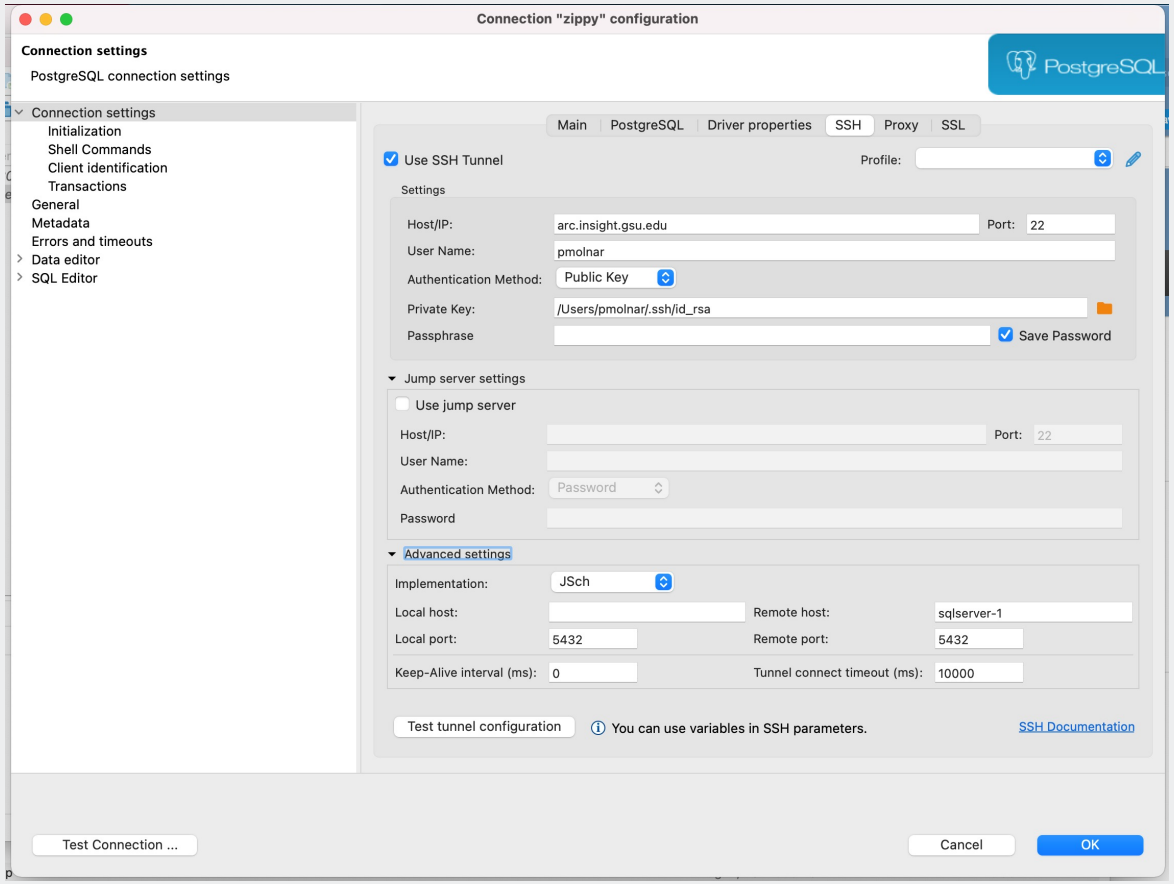
Free to use desktop applications:

- DBeaver Community
  https://dbeaver.io
  (Mac/Windows/Linux)

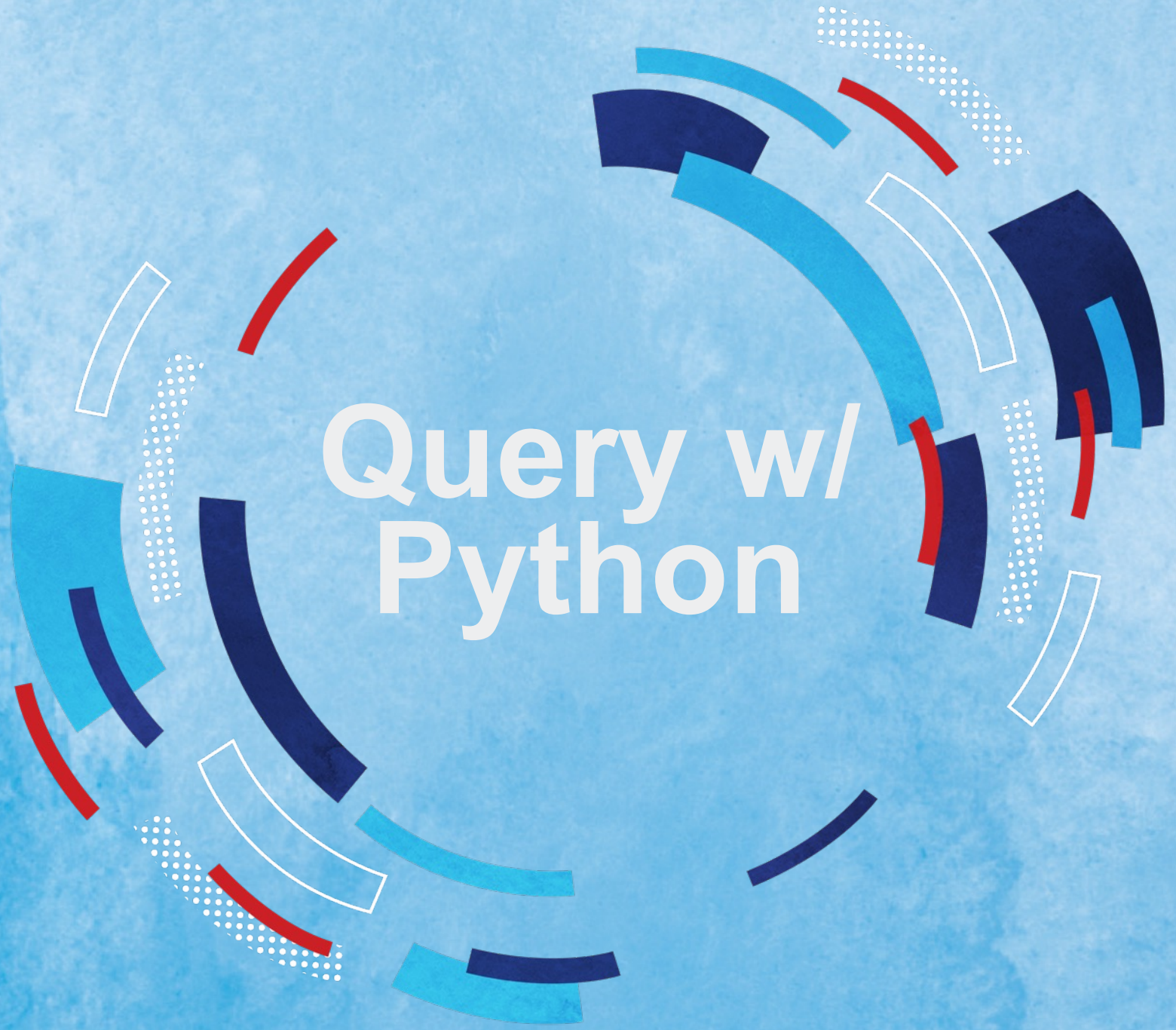
- SQL Workbench/J
  https://www.sql-workbench.eu
  (Java)

SQLworkbench

Connection via SSH Tunnel



Query w/ Python



Georgia State University | J. Mack Robinson College of Business

Robinson

Programmatic access to SQL Databases

```
import os
import json
import sys
import re
import numpy as np
import pandas as pd
import sqlalchemy as sa
```

```
# load database credentials and change database to "datalabeling"
db_creds = json.load(open(f"/home/{os.environ['USER']}/"
                          ".arc_config/databases/sqlserver-2-3306.txt"))
connection_uri = '/'.join( db_creds['default_uri'] \
                          .split('/')[:3] + ['datalabeling'])

# create connection
conn = sa.engine.create_engine(connection_uri)

# test if connection works
q = """
SELECT NOW() AS jetzt
"""
display(pd.read_sql(q, conn))
```

	jetzt
0	2021-04-09 13:11:50

Query SQL with Pandas

```
user_id = 'xfan2'
q = f"""
SELECT COUNT(a.assignment_id) as n_completed
  FROM licensing_assignments a
  JOIN licensing_responses r
  ON a.assignment_id=r.assignment_id
  WHERE a.user_id='{user_id}'
"""
pd.read_sql(q, con=conn)
```

	n_completed
0	5

Limits

Limitations

The database servers on ARC

Database systems are for development and prototyping only. They do not support large scale data and analysis

Default: one database per user or project

Number of tables in database is not limited (beyond system limits)

Database owners can grant access to other ARC users. Creating additional users is not allowed

Users are responsible to maintain data backups in case the database gets

Request non-person logins for applications

SQL Exercises

https://github.com/kingmolnar/advanced-sql-for-data-scientists

Distributed Computing

Introduction to the Analytics Research Cluster (ARC)

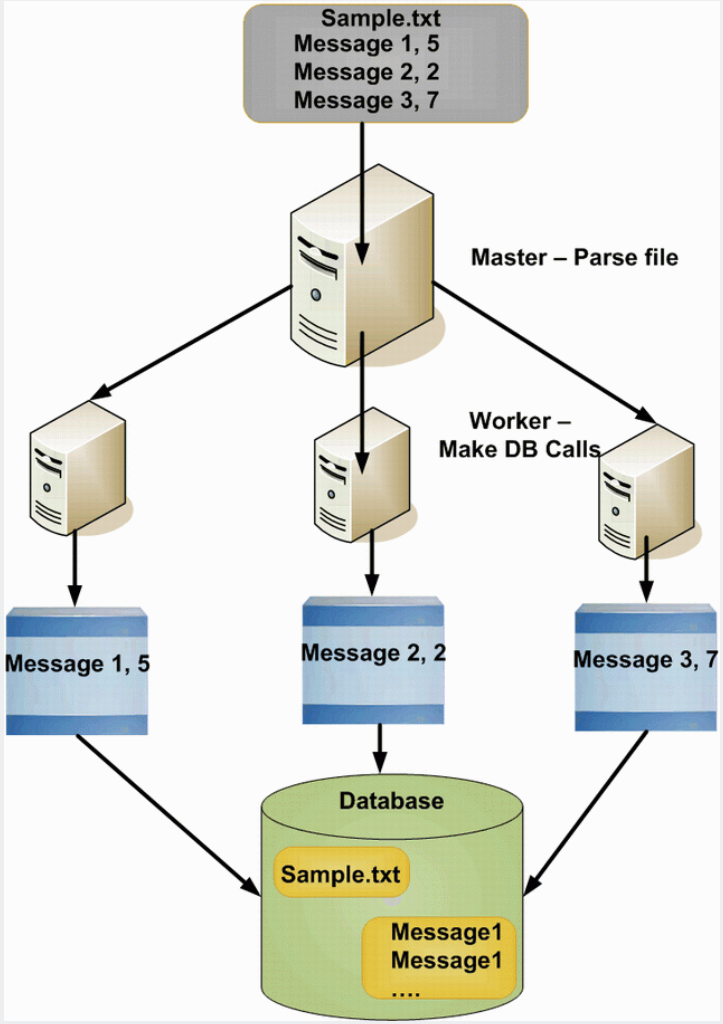
October 21, 2023

Georgia State University | J. Mack Robinson College of Business

Robinson

Outline

- Parallel Processing
- Map Reduce
- Apache Spark
- Other Frameworks



Apache Spark

Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine

Data structure for representation of **tables** and **rows**

Lazy evaluation: operations are only executed when required, define transformations for all records, evaluate when needed

Combine external distributed storage and in-memory processing

Benefits:

- Runs locally and on large cluster, same code to scale up
- Parallel processing, fast and scalable

Challenges:

- Additional step for software installation required

https://spark.apache.org/docs/latest/

Pandas map() and apply()

The methods `map()` and `apply()` can be used for row and column operations:

map(self, arg, na_action=None) -> 'Series'

Map values of Series according to input correspondence.

```
Used for substituting each value in a Series with another value,
that may be derived from a function, a ``dict`` or
a :class:`Series`.

Parameters
----------
arg : function, collections.abc.Mapping subclass or Series
    Mapping correspondence.
na_action : {None, 'ignore'}, default None
    If 'ignore', propagate NaN values, without passing them to the
    mapping correspondence.

Returns
-------
Series
    Same index as caller.
```

apply(self, func, convert_dtype=True, args=(), **kwds)

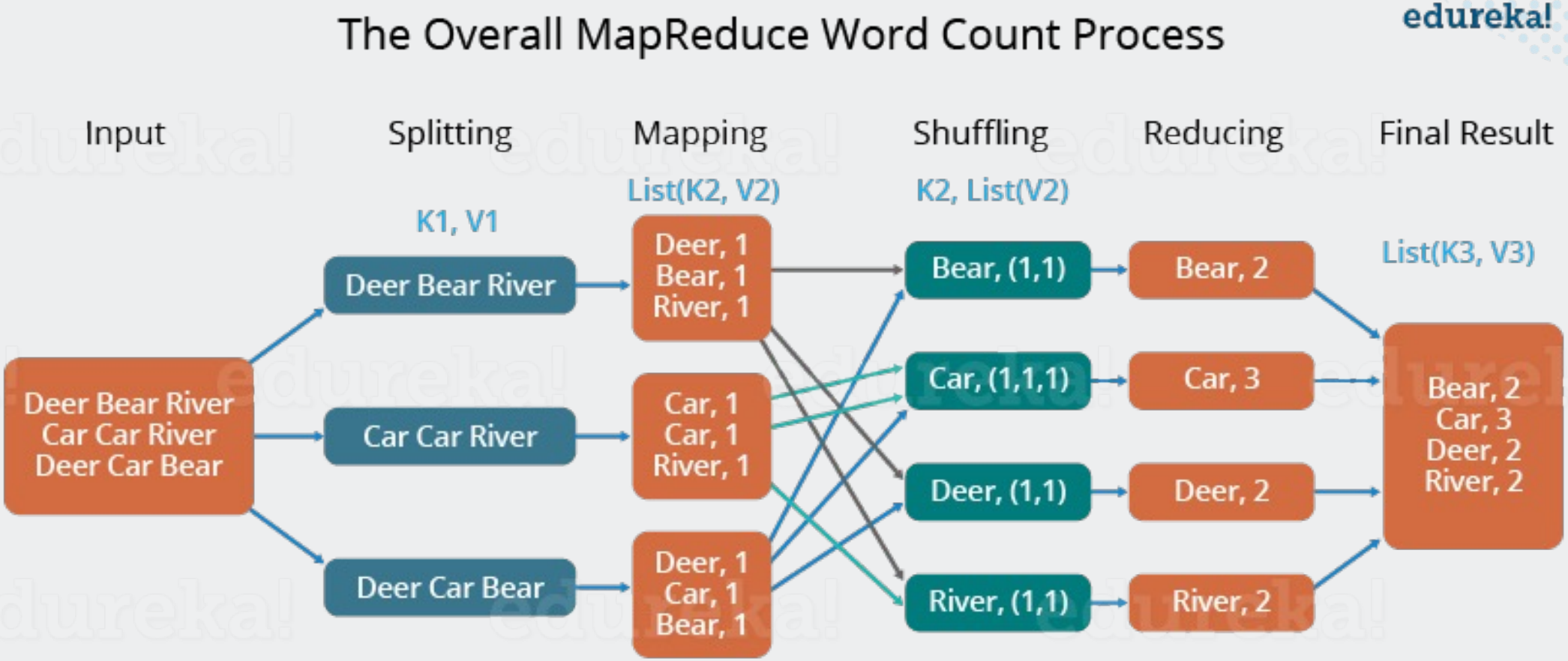
Invoke function on values of Series.

```
Can be ufunc (a NumPy function that applies to the entire Series)
or a Python function that only works on single values.

Parameters
----------
func : function
    Python function or NumPy ufunc to apply.
convert_dtype : bool, default True
    Try to find better dtype for elementwise function results. If
    False, leave as dtype=object.
args : tuple
    Positional arguments passed to func after the series value.
**kwds
    Additional keyword arguments passed to func.

Returns
-------
Series or DataFrame
    If func returns a Series object the result will be a DataFrame.
```

Map-Reduce & Group By



Word Count Example

Python Pandas

```
from functools import reduce

pdf = pd.read_csv('three_by_three.csv')

pdf2 = pd.DataFrame({
          'word': reduce(
             lambda a, b: a + b,
             pdf.text.str.split()
          )
         })
pdf2['n'] = 1

pdf2.groupby('word').agg({'n': 'sum'})
```

SQL

```
SELECT word, count(1) AS n
FROM (
    SELECT explode(split(text, ' ')) AS
word
    FROM threebythree
)
GROUP BY word
```

Apache Spark

```
df.select(
   F.explode(
      F.split('text', r'\s')
   ).alias('word')) \
   .groupBy('word').agg(
      F.count('word').alias('n')
   ) \
   .show()
```

Joining Tables: SQL

Query classic and vintage car sales that were sold by US sales reps?

Employees with job title "Sales Rep" who are assigned to an office in the US and are the sales rep for customers of vintage and classic cars.

Using Common Table Expressions (CTEs) to structure the statements.

Select only required fields to avoid duplicates.

```
WITH us_sales_reps AS (
    SELECT employeeNumber, lastName,
           firstName, city, postalCode
    FROM employees e
    JOIN offices o
        ON e.officeCode = o.officeCode
    WHERE e.jobTitle = 'Sales Rep' AND o.country = 'USA'
),
car_orders AS (
    SELECT prod.productCode, prod.buyPrice,
           prod.MSRP, det.priceEach, det.quantityOrdered,
           or.customerNumber, or.shippedDate
    FROM orders or
    JOIN orderdetails det ON or.orderNumber = det.orderNumber
    JOIN products prod ON det.productCode = prod.productCode
    WHERE prod.productLine in ('Classic Cars', 'Vintage Cars') AND or.status = 'Shipped'
)
SELECT sr.*, car.*
FROM car_orders car
JOIN customers cst ON car.customerNumber = cst.customerNumber
JOIN us_sales_reps sr
    ON cst.salesRepEmployeeNumber = sr.employeeNumber
```

Joining Tables: Pandas

Query classic and vintage car sales that were sold by US sales reps?

Employees with job title "Sales Rep" who are assigned to an office in the US and are the sales rep for customers of vintage and classic cars.

Filtering cannot be chained to the same expression. Require multiple assignments.

Select only required fields to avoid duplicates.

```
df1 = pd.merge(employees_pdf, offices_pdf, on='officeCode')
df2 = df1[ (df1.jobTitle == 'Sales Rep') & (df1.country == 'USA') ]
us_sales_reps_pdf = df2[['employeeNumber', 'lastName', 'firstName', 'city']]

df3 = pd.merge(
        pd.merge(orders_pdf, orderdetails_pdf, on='orderNumber'),
        products_pdf, on='productCode'
      )
df4 = df3[ df3.productLine.isin(['Classic Cars', 'Vintage Cars'])
                                              & (df3.status == 'Shipped') ]
car_orders_pdf = df4[['productCode', 'buyPrice', 'MSRP', 'priceEach', 'quantityOrdered',
                      'customerNumber', 'shippedDate']]

df5 = pd.merge(
    us_sales_reps_pdf,
    pd.merge(car_orders_pdf, customers_pdf[['customerNumber', 'salesRepEmployeeNumber']],
on='customerNumber'),
    left_on='employeeNumber', right_on='salesRepEmployeeNumber'
)
us_car_orders_pdf = df5[['employeeNumber', 'lastName', 'firstName', 'city',
                         'productCode', 'productCode', 'buyPrice', 'MSRP', 'priceEach',
                         'quantityOrdered', 'customerNumber', 'shippedDate']]
```

Joining Tables: Spark

Query classic and vintage car sales that were sold by US sales reps?

Employees with job title "Sales Rep" who are assigned to an office in the US and are the sales rep for customers of vintage and classic cars.

Closely resembles SQL query. More flexibility to control order of expressions.

Select only required fields to avoid duplicates.

```
us_sales_reps_df = employees_df \
    .join(offices_df, 'officeCode') \
    .where("""jobTitle = 'Sales Rep' AND country = 'USA'""") \
    .select('employeeNumber', 'lastName', 'firstName', 'city')

car_orders_df = orderdetails_df \
    .join(products_df, 'productCode') \
    .where("productLine in ('Classic Cars', 'Vintage Cars')") \
    .join(orders_df, 'orderNumber') \
    .where("status = 'Shipped'") \
    .select(
          'productCode', 'buyPrice', 'MSRP', 'priceEach',
          'quantityOrdered', 'customerNumber', 'shippedDate'
     )

us_car_orders_df = customers_df \
    .select('customerNumber', 'salesRepEmployeeNumber') \
    .join(us_sales_reps_df,
              us_sales_reps_df.employeeNumber == customers_df.salesRepEmployeeNumber
     ) \
    .join(car_orders_df, 'customerNumber')
```

Launch PySpark Notebooks

On ARC

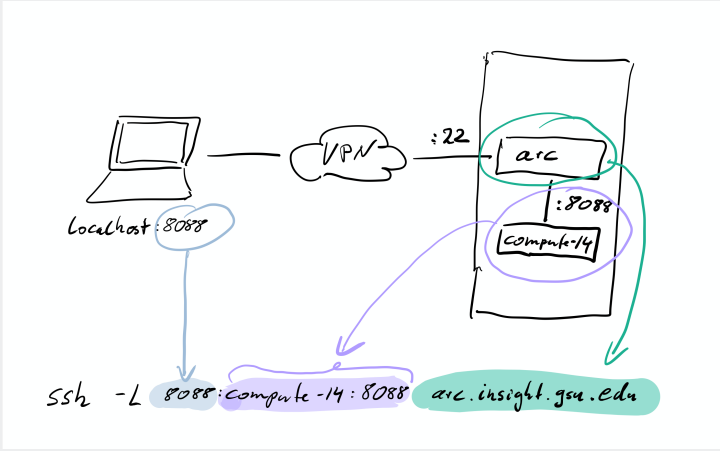
```
[arc ~]$ export PYSPARK_DRIVER_PYTHON=jupyter
[arc ~]$ export PYSPARK_DRIVER_PYTHON_OPTS='notebook --no-browser'
[arc ~]$ pyspark
To access the notebook, open this file in a browser:
        file:///home/pmolnar/.local/share/jupyter/runtime/nbserver-323269-open.html
    Or copy and paste one of these URLs:
        http://localhost:8889/?token=3b8147e4a168a080be86d6ac50375c9501f4b85a10666ba7
     or http://127.0.0.1:8889/?token=3b8147e4a168a080be86d6ac50375c9501f4b85a10666ba7
```

Port **number** and **token** are different every time

On local computer (laptop, desktop)

```
[macos ~]$ ssh -fNL 8889:localhost:8889 arc.insight.gsu.edu
[macos ~]$ ssh -fNL 8088:compute-14:8088 arc.insight.gsu.edu
```

Open web-browser at http://localhost:8889, and http://localhost:8088



Run Spark Jobs

Submit Job

```
[arc ~]$ spark-submit --master yarn \
                    -- ... \
                    myprog.py OPT1 OPT2 OPT3
```

Check Job Status

1. Command Line Interface, check job queue

```
[arc ~]$ yarn top
```

2. Web-interface:
   a. Connect browser to these services: http://HOST:PORT
   b. Tunnel with:

```
ssh -fNL LOCALPORT:HOST:PORT \
        user@arc.insight.gsu.edu
```

Service	Host	Port
Yarn Manager and Spark UI Proxy	compute-14	8088
Spark History Server	compute-14	18080
PostgreSQL Server	sqlserver-1	5432

Other Frameworks

Do not run parallel workflows on the head node!

DASK

Dask uses existing Python APIs and data structures to make it easy to switch between NumPy, pandas, scikit-learn to their Dask-powered equivalents.
https://dask.org

RAY

Support workloads like deep learning and hyperparameter tuning, that are compute-intensive, and require distributed or parallel execution.
https://www.ray.io

Joblib

Joblib is a set of tools to provide lightweight pipelining in Python. Supports transparent disk-caching of functions and lazy re-evaluation (memoize pattern).
https://joblib.readthedocs.io