

# SINGLE-NUCLEI RNA-SEQ ANALYSIS USING AMAZON WEB SERVICES (AWS)

Izar Lab — October 2020

Jana Biermann, PhD  
[jb4424@cumc.columbia.edu](mailto:jb4424@cumc.columbia.edu)

Yiping Wang, PhD  
[yipingwang12@gmail.com](mailto:yipingwang12@gmail.com)

# Overview

1. Course preparation
2. AWS introduction
3. Launching an instance
4. SSH into your instance
5. Updating the instance and installing AWS CLI
6. Installing Cell Ranger
7. Creating a snRNA-seq reference
8. Creating, attaching and mounting a volume
9. Downloading raw data
10. Running Cell Ranger
11. Setting up an alarm for your instance
12. Running CellBender
13. Running Seurat

# COURSE PREPARATION

# Course preparation: Installing OSXfuse

FUSE allows you to mount and interact with directories and files located on a remote server over a normal ssh connection

- Go to <https://osxfuse.github.io/>
- Download FUSE and SSHFS
- Open the Terminal and check if it's working:  

```
sshfs --version
```
- If you can see the version number, the installation was successful
- Windows users can install MobaXterm to access remote servers using SSH; a graphical SFTP browser is included (<https://mobaxterm.mobatek.net>)

**FUSE for macOS**  
File system integration made easy

Project on GitHub Downloads Wiki Google Group Issue Tracker

What is FUSE for macOS?

FUSE for macOS allows you to extend macOS's native file handling capabilities via third-party file systems. It is a successor to MacFUSE, which has been used as a software building block by dozens of products, but is no longer being maintained.

Features

As a user, installing the FUSE for macOS software package will let you use any third-party FUSE file system. Legacy MacFUSE file systems are supported through the optional MacFUSE compatibility layer.

As a developer, you can use the FUSE SDK to write numerous types of new file systems as regular user space programs. The content of these file systems can come from anywhere: from the local disk, from across the network, from memory, or any other combination of sources. Writing a file system using FUSE is orders of magnitude easier and quicker than the traditional approach of writing in-kernel file systems. Since FUSE file systems are regular applications (as opposed to kernel extensions), you have just as much flexibility and choice in programming tools, debuggers, and libraries as you

**Stable Releases**

FUSE for macOS 3.11.0 Mac OS X 10.5 or later Intel or PowerPC Released on 04 Jul 2020 Downloaded 132,675 times
SSHFS 2.5.0 Mac OS X 10.5 or later Intel or PowerPC Released on 03 Feb 2014 Downloaded 746,370 times

Recent Posts Archive

Release of FUSE for macOS 3.11.0  
Posted on 04 Jul 2020

Release of FUSE for macOS 3.10.6  
Posted on 08 Jun 2020

# Course preparation: Setting up R and installing libraries

- Follow steps 1-4 on this web page to install R, RStudio and the Swirl R-package:  
<https://swirlstats.com/students.html>
- Please work through the following Swirl courses before the beginning of the workshop (depending on your familiarity with R):
  - 1: R Programming: The basics of programming in R
  - 4: Exploratory Data Analysis: The basics of exploring data in R
- Install the following libraries before the beginning of the workshop and make sure you can access them by calling `library(nameoflibrary)` without any errors:
  - `install.packages(c('dplyr', 'BiocManager', 'httr', 'plotly', 'Seurat', 'cowplot', 'ggplot2', 'gplots', 'ggthemes', 'stringr', 'pheatmap', 'Matrix'))`
  - `BiocManager::install(c('SingleR', 'SingleCellExperiment', 'scater', 'purrr', 'celldex'))`

# Setting up Python and associated libraries

- Install Python 3 by running:
  - sudo apt-get install python3
- Install pip, a package manager that helps with installing libraries on python, by running:
  - sudo apt-get install python3-pip
- Install scrublet, a python package for detecting doublets, by running:
  - pip3 install scrublet
- While running the above command, you may encounter an error about a library called llvm missing. To fix this, run the below commands, and then rerun pip3:
  - sudo bash -c "\$(wget -O - <https://apt.llvm.org/llvm.sh>)"
  - sudo ln -s /usr/bin/llvm-config-10 /usr/bin/llvm-config
  - LLVM\_CONFIG=/usr/bin

# Course preparation: Command line tutorials

- Work through any of these free command line tutorials (as you see fit):

<https://blog.teamtreehouse.com/introduction-to-the-mac-os-x-command-line>

<https://www.udacity.com/course/linux-command-line-basics--ud595>

<https://www.udemy.com/course/command-line/>

<https://ubuntu.com/tutorials/command-line-for-beginners#1-overview>

- You should be familiar with basic commands such as pwd, ls, cd, mv, cp, mkdir etc.

# Course preparation: Sign in to AWS

- When your AWS account is created you'll receive a credentials files that should be kept in a hidden folder
- Try signing in to AWS to make sure it's working before the beginning of the course
- You'll find your one time-use password in the credentials file

<https://us-east-2.signin.aws.amazon.com>



## Sign in as IAM user

Account ID (12 digits) or account alias

IAM user name

Password

Remember this account

**Sign in**

# Course preparation: Reading list

- Please read the following papers and pre-prints before the beginning of the course:
  - Current best practices in single-cell RNA-seq analysis: a tutorial
    - [10.15252/msb.20188746](https://doi.org/10.15252/msb.20188746)
  - CellBender remove-background: a deep generative model for unsupervised removal of background noise from scRNA-seq datasets
    - <https://doi.org/10.1101/791699>
  - Scrublet: Computational Identification of Cell Doublets in Single-Cell Transcriptomic Data
    - [10.1016/j.cels.2018.11.005](https://doi.org/10.1101/10.1016/j.cels.2018.11.005)

# Important points

- You can find your password in the credentials file
- Keep the credentials file save in a hidden folder
- S3 is in most cases the only copy of our data so make sure not to delete or alter anything
- Most people in the lab have access to S3 so you can direct them to your analysis output
- Downloading data from AWS is very expensive so please don't download large files but instead analyze them on AWS instances until you reach a smaller format that can be downloaded
- The volume storage associated with instances is also very expensive so try to keep it small and store data on S3
- Please only use your own instances, volumes and images generated with your own key
- If you don't need an instance anymore you can terminate it
- You can check the budget in the 'Cost Explorer' menu and filter for costs generated with your key. This will only include instance-related costs and not the costs generated through S3 and volumes.

# INTRODUCTION TO AWS

# Cloud computing with AWS

- On-demand availability of computer system resources, especially data storage and computing power
- Based on pay-as-you-go concept
- AWS technology is implemented at server farms throughout the world
- Amazon Elastic Compute Cloud (EC2):
  - Called an "instance"
  - Rent virtual computers to run your own applications
  - Boot Amazon Machine Image (AMI) to configure a virtual machine containing specific software
  - Create, launch, and terminate server-instances as needed
- Emulates most attributes of real computers:
  - Hardware central processing units (CPUs)
  - Graphics processing units (GPUs)
  - Local/RAM memory
  - Hard-disk/SSD storage
  - Choice of operating systems
  - Networking etc.

# AWS Global Infrastructure



- The AWS Global Infrastructure is designed and built to deliver a flexible, reliable, scalable, and secure cloud computing environment with high-quality global network performance.
- This map shows the current AWS Regions and more that are coming soon.



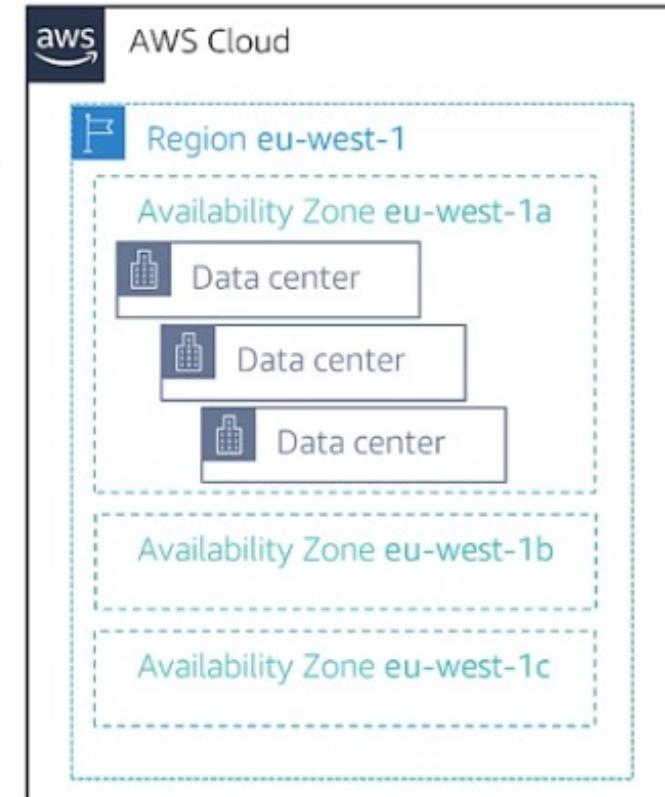
© 2020 Amazon Web Services, Inc. or its Affiliates. All rights reserved.

10

# Availability Zones



- Each **Region** has multiple Availability Zones.
- Each **Availability Zone** is a fully isolated partition of the AWS infrastructure.
  - There are currently 69 Availability Zones worldwide
  - Availability Zones consist of discrete **data centers**
  - They are designed for fault isolation
  - They are interconnected with other Availability Zones by using high-speed private networking
  - You choose your Availability Zones.
  - AWS recommends replicating data and resources across Availability Zones for resiliency.



- ❖ You need to stay within the region us-east-2 (Ohio)
- ❖ Availability Zones can be freely chosen

# Sign in

<https://us-east-2.signin.aws.amazon.com>



## Sign in as IAM user

Account ID (12 digits) or account alias

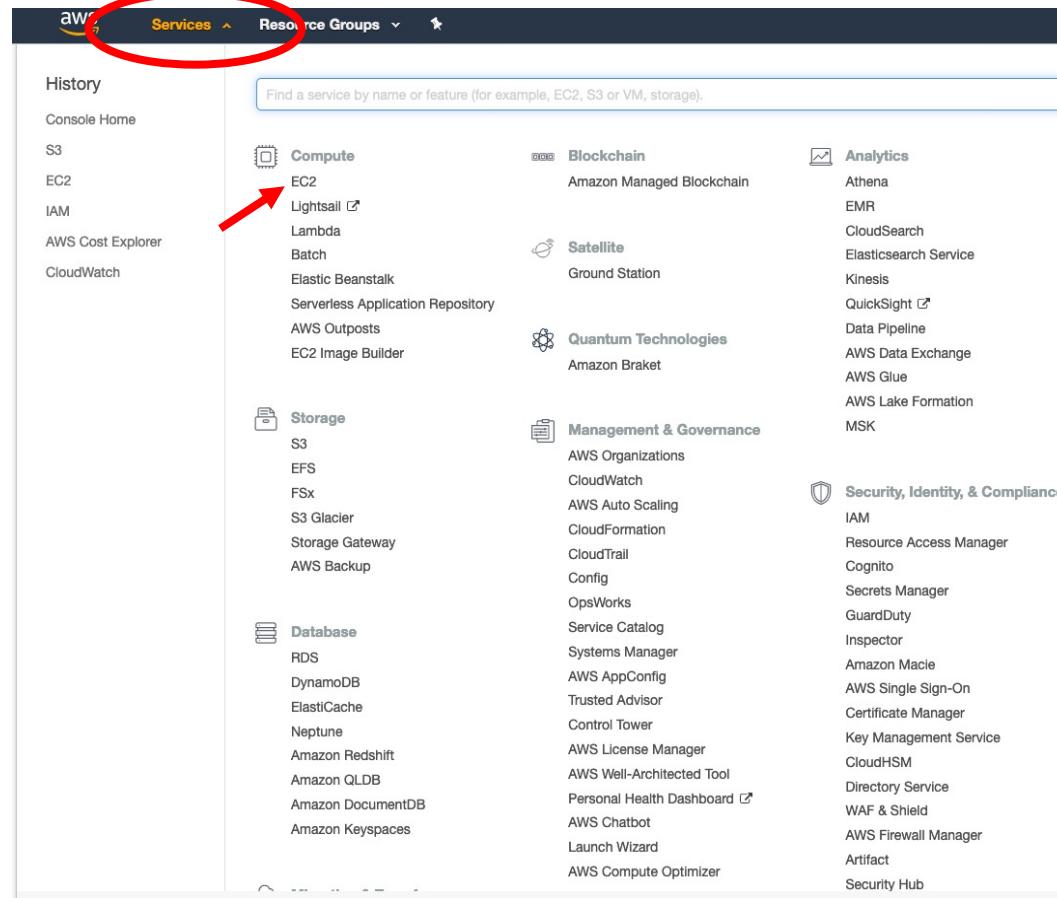
IAM user name

Password

Remember this account

**Sign in**

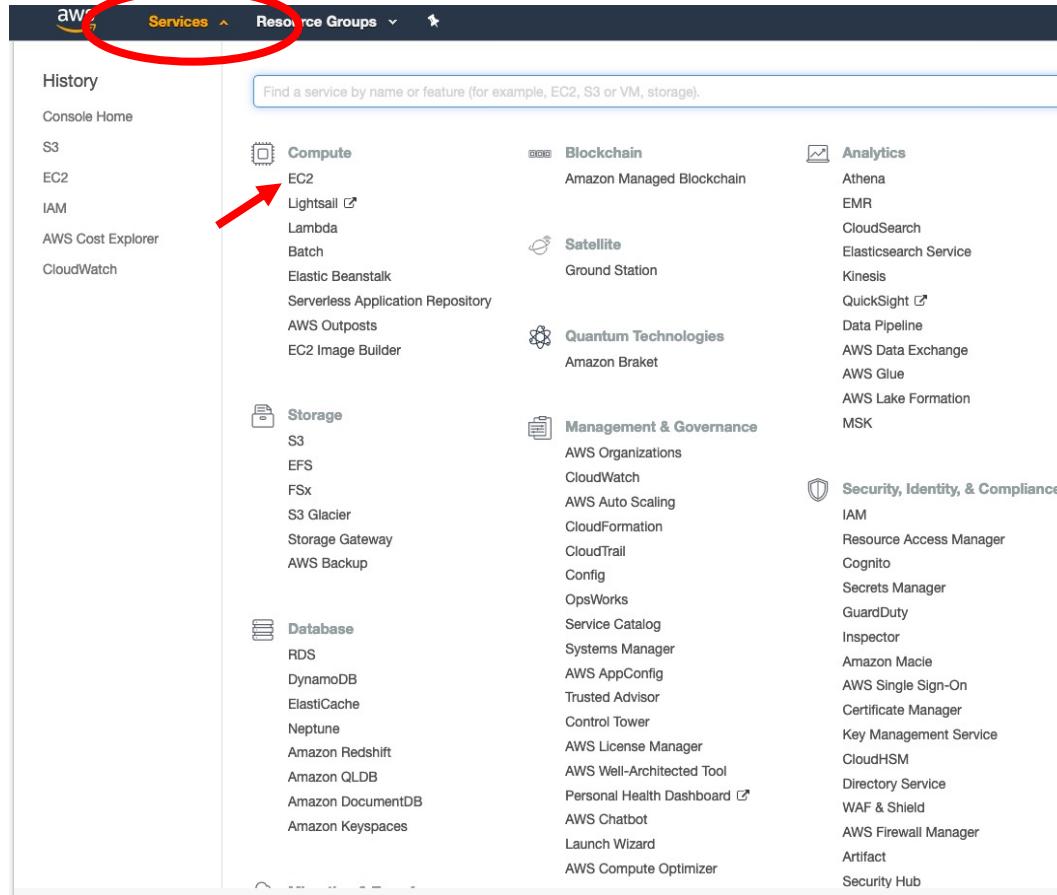
# Navigate to Key Pairs and create your key



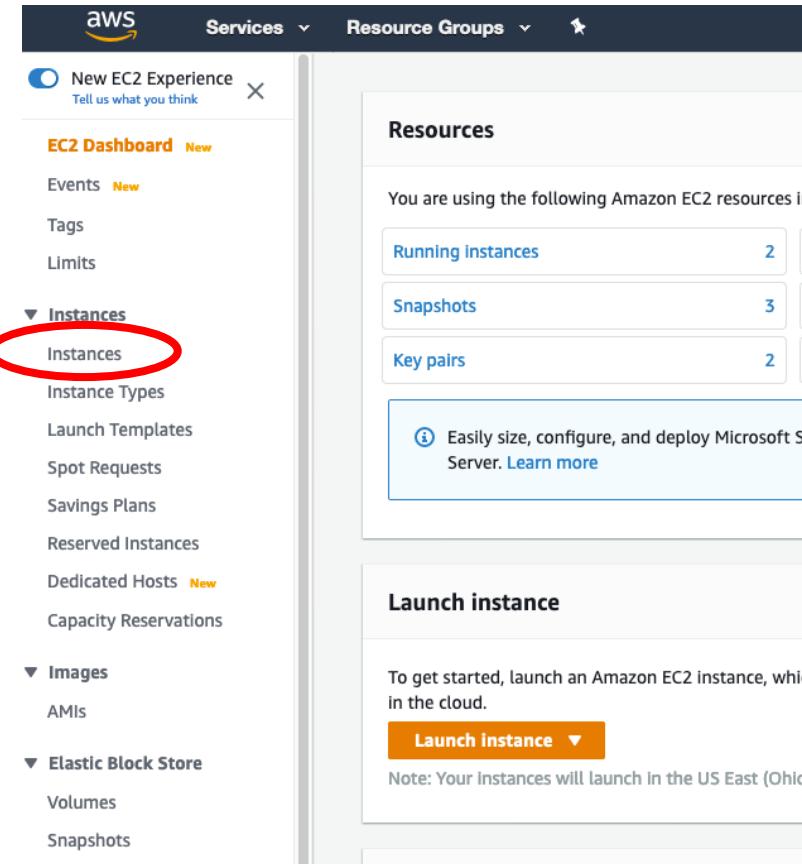
- ❖ Name your key starting with your name\_key
- ❖ E.g. 'yiping\_key'

A screenshot of the EC2 Dashboard. The 'Key Pairs' link is circled in red. The dashboard includes sections for Instances, Images, Elastic Block Store, Network &amp; Security, and a sidebar for Resources.

# Navigate to E2/Instances



The screenshot shows the AWS Management Console navigation bar at the top. Below it is a search bar with placeholder text: "Find a service by name or feature (for example, EC2, S3 or VM, storage)". On the left, there's a sidebar with links to History, Console Home, S3, EC2, IAM, AWS Cost Explorer, and CloudWatch. The main area is titled "Compute" and contains a list of services: EC2 (with a red arrow pointing to it), Lightsail, Lambda, Batch, Elastic Beanstalk, Serverless Application Repository, AWS Outposts, EC2 Image Builder, Storage (S3, EFS, FSx, S3 Glacier, Storage Gateway, AWS Backup), Database (RDS, DynamoDB, ElastiCache, Neptune, Amazon Redshift, Amazon QLDB, Amazon DocumentDB, Amazon Keyspaces), and Analytics (Amazon Managed Blockchain, Athena, EMR, CloudSearch, Elasticsearch Service, Kinesis, QuickSight, Data Pipeline, AWS Data Exchange, AWS Glue, AWS Lake Formation, MSK). Below these are sections for Management & Governance (AWS Organizations, CloudWatch, AWS Auto Scaling, CloudFormation, Config, OpsWorks, Service Catalog, Systems Manager, AWS AppConfig, Trusted Advisor, Control Tower, AWS License Manager, AWS Well-Architected Tool, Personal Health Dashboard, AWS Chatbot, Launch Wizard, AWS Compute Optimizer) and Security, Identity, & Compliance (IAM, Resource Access Manager, Cognito, Secrets Manager, GuardDuty, Inspector, Amazon Macie, AWS Single Sign-On, Certificate Manager, Key Management Service, CloudHSM, Directory Service, WAF & Shield, AWS Firewall Manager, Artifact, Security Hub).



The screenshot shows the AWS EC2 Dashboard. At the top, there's a toggle for "New EC2 Experience" with a "Tell us what you think" link. Below that is a "Resources" section with a message: "You are using the following Amazon EC2 resources in". It lists "Running Instances" (2), "Snapshots" (3), and "Key pairs" (2). A callout box says: "Easily size, configure, and deploy Microsoft SQL Server. Learn more". The main content area is titled "Launch instance" with a sub-section "To get started, launch an Amazon EC2 instance, which in the cloud." A large orange "Launch instance" button is prominently displayed. Below it, a note says: "Note: Your instances will launch in the US East (Ohio)". The dashboard also includes sections for "Instances" (with a red circle around it), "Instance Types", "Launch Templates", "Spot Requests", "Savings Plans", "Reserved Instances", "Dedicated Hosts", "Capacity Reservations", "Images" (AMIs), and "Elastic Block Store" (Volumes, Snapshots).

# Launching an instance for CellRanger

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various navigation links like 'EC2 Dashboard', 'Tags', 'Limits', 'Instance Types', 'Launch Templates', etc. A large red box highlights the 'Insta' link in the sidebar, with the text 'Create a new instance here' overlaid. At the top, there's a blue 'Launch Instance' button with a red arrow pointing to it. Below the button is a search bar and a table listing existing instances. The table columns include Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, and Alarm Status. Some instances are labeled with names starting with 'a\_'. The bottom of the page has a note 'Select an instance above'.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
a_cellbender	i-0cac179fe713b5fba	p2.xlarge	us-east-2a	stopped	No Data	
a_cellbender2_image	i-0e7c8dcc56e805210	p2.xlarge	us-east-2a	stopped	No Data	
a_cellranger	i-04e2e6f345bba9ede	r5.4xlarge	us-east-2a	stopped	No Data	
a_cellranger2_image	i-0fcade29e5f681dd3	r5.24xlarge	us-east-2a	stopped	No Data	
jana_inferCNV	i-0bdd236fa1f93c94f	r5.12xlarge	us-east-2a	stopped	No Data	
jana_seurat	i-027816fb750c550ea	r5.2xlarge	us-east-2a	stopped	No Data	
jana_seurat2	i-09ad34637dd8a403f	r5.4xlarge	us-east-2a	stopped	No Data	
yiping_instance2	i-0037066438d768fda	t2.medium	us-east-2a	running	2/2 checks ...	None
yiping_instance_large	i-0111cac05d48d23dd	r5.4xlarge	us-east-2a	running	2/2 checks ...	None

- ❖ Name your instances and volumes starting with your name
- ❖ E.g. 'yiping\_instance\_large'

# Launching an instance — Step 1

1. Choose AMI    2. Choose Instance Type    3. Configure Instance    4. Add Storage    5. Add Tags    6. Configure Security Group    7. Review

**Step 1: Choose an Amazon Machine Image (AMI)**

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows" Cancel and Exit

Search by Systems Manager parameter

Quick Start

Category	AMI Name	Description	Select
My AMIs	Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-07c8bc5c1ce9598c3 (64-bit x86) / ami-09a67037138f86e67 (64-bit Arm)	Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.	<input checked="" type="button"/> Select
AWS Marketplace	Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-02b0c55eeae6d5096	The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.	<input type="button"/> Select
Community AMIs	Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0a54aef4ef3b5f881 (64-bit x86) / ami-0ffd59b53e6797671 (64-bit Arm)	Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type	<input checked="" type="button"/> Select
Free tier only	SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-03f4c416f489586a3 (64-bit x86) / ami-0d24f1c1ba96d2803 (64-bit Arm)	SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.	<input checked="" type="button"/> Select
	Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0bbe28eb2173f6167 (64-bit x86) / ami-04adf33460efc8798 (64-bit Arm)	Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical ( <a href="http://www.ubuntu.com/cloud/services">http://www.ubuntu.com/cloud/services</a> ).	<input checked="" type="button"/> Select

1 to 40 of 40 AMIs

**Choose Ubuntu**

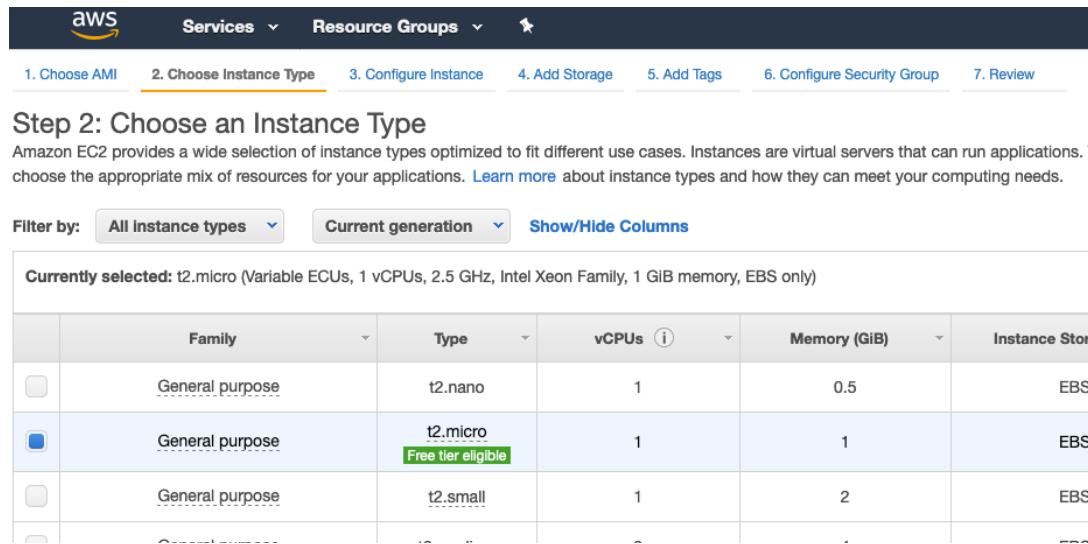
Free tier eligible

Free tier eligible

Free tier eligible

Free tier eligible

# Launching an instance — Step 2



Step 2: Choose an Instance Type

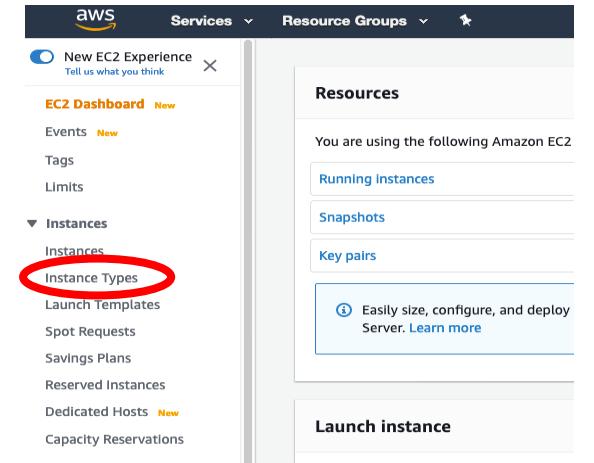
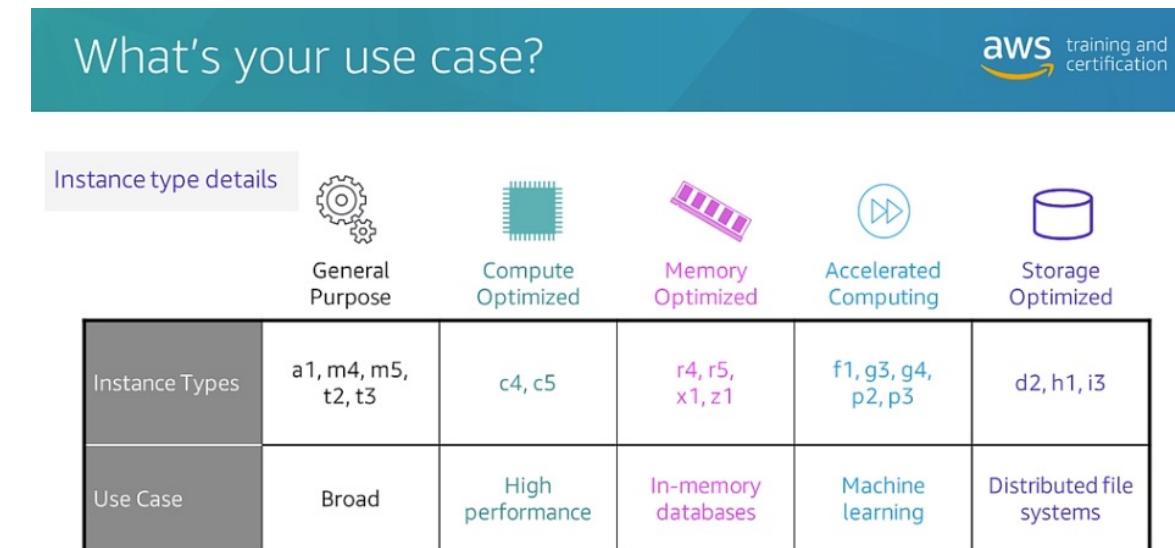
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. To choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All Instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

Family	Type	vCPUs	Memory (GiB)	Instance Storage
General purpose	t2.nano	1	0.5	EBS only
General purpose	<b>t2.micro</b> Free tier eligible	1	1	EBS only
General purpose	t2.small	1	2	EBS only

- ❖ Choose an instance based on use case
- ❖ Check properties of different instances in the ‘Instance Types’ menu
- ❖ Requirements for Cell Ranger (one sample):
  - ❖ 16 cores
  - ❖ 128 GB
- Select instance r5.4xlarge here



© 2020 Amazon Web Services, Inc. or its Affiliates. All rights reserved.

New EC2 Experience Tell us what you think

EC2 Dashboard New

Events New

Tags

Limits

Instances

Instance Types (highlighted)

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts New

Capacity Reservations

Resources

You are using the following Amazon EC2

Running Instances

Snapshots

Key pairs

Easily size, configure, and deploy Server. [Learn more](#)

Launch instance

# Launching an instance — Step 3

1. Choose AMI   2. Choose Instance Type   3. Configure Instance   4. Add Storage   5. Add Tags   6. Configure Security Group   7. Review

## Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of Instances  Launch into Auto Scaling Group

Purchasing option  Request Spot instances

Network   Create new VPC

Subnet

Auto-assign Public IP

Placement group  Add instance to placement group

Capacity Reservation

IAM role   Create new IAM role

Shutdown behavior

Stop - Hibernate behavior  Enable hibernation as an additional stop behavior

Enable termination protection  Protect against accidental termination

Monitoring  Enable CloudWatch detailed monitoring  
Additional charges apply.

Tenancy   
Additional charges will apply for dedicated tenancy.

Elastic Inference  Add an Elastic Inference accelerator  
Additional charges apply.

T2/T3 Unlimited  Enable  
Additional charges may apply

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Storage](#)

# Launching an instance — Step 4

1. Choose AMI   2. Choose Instance Type   3. Configure Instance   4. Add Storage   5. Add Tags   6. Configure Security Group   7. Review

## Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

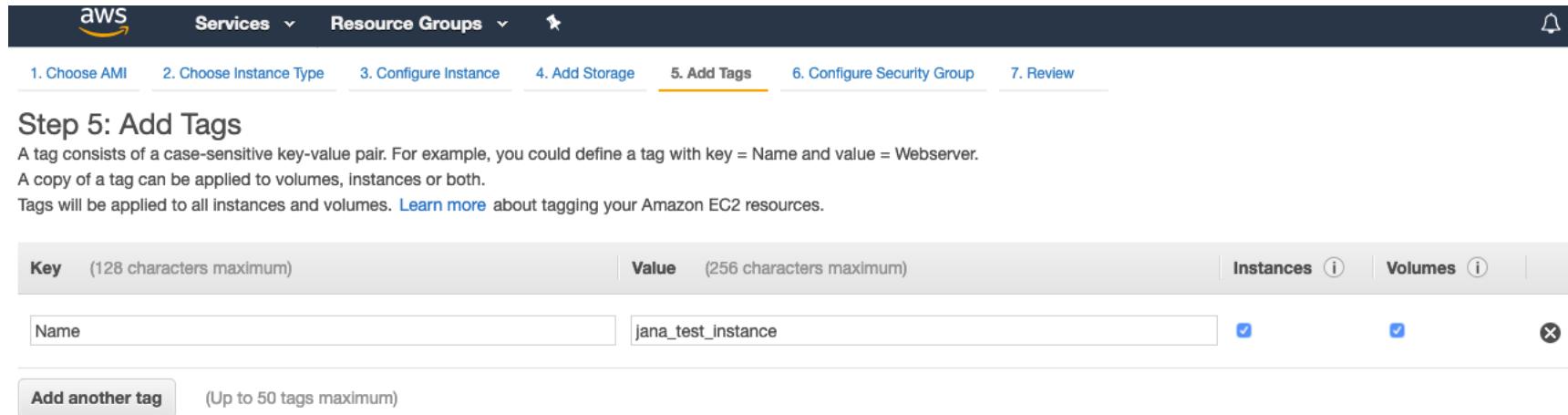
Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-0cd98f931a8fffac8	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

**Add New Volume**

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

- ❖ Choose the size of the root volume
  - ❖ The size can also be changed later in the ‘Volumes’ menu (only to increase it and it takes some time)
  - ❖ You can add additional volumes to your instance (need to be attached and then mounted)
  - ❖ This type of storage is expensive
- Choose 50 GiB today

# Launching an instance — Step 5



The screenshot shows the AWS EC2 instance launch wizard at Step 5: Add Tags. The top navigation bar includes the AWS logo, Services dropdown, Resource Groups dropdown, and a bell icon. Below the navigation, a progress bar shows steps 1 through 7: Choose AMI, Choose Instance Type, Configure Instance, Add Storage, Add Tags (highlighted in yellow), Configure Security Group, and Review. The main content area is titled "Step 5: Add Tags". It explains that a tag consists of a case-sensitive key-value pair, such as "Name = Webserver". It also notes that a copy of a tag can be applied to volumes, instances or both, and that tags will be applied to all instances and volumes. A link to "Learn more" about tagging is provided. The "Instances" tab is selected, showing a single tag named "Name" with the value "jana\_test\_instance". There are checkboxes for "Instances" and "Volumes", both of which are checked. A red "X" button is available to remove the tag. At the bottom left is a "Add another tag" button with the note "(Up to 50 tags maximum)".

- ❖ Name your instances and volumes starting with your name
- ❖ E.g. 'yiping\_cellranger'

# Launching an instance — Step 6

The screenshot shows the AWS Launch Wizard Step 6: Configure Security Group. At the top, there are tabs for 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group (which is highlighted), and 7. Review. Below the tabs, the heading "Step 6: Configure Security Group" is displayed. A descriptive text explains that a security group is a set of firewall rules that control the traffic for your instance. It mentions that you can add rules to allow specific traffic to reach the instance and that it allows unrestricted access to the HTTP and HTTPS ports. There is a link to learn more. Below this, there is a section titled "Assign a security group:" with two options: "Create a new security group" (radio button) and "Select an existing security group" (radio button, which is selected). A table lists existing security groups:

Security Group ID	Name	Description
sg-e9c01990	default	default VPC security group
sg-01bb301cd3293a98b	launch-wizard-1	launch-wizard-1 created 2020-04-03T11:40:11Z
sg-09f744ffee14e8cb7	launch-wizard-2	launch-wizard-2 created 2020-07-13T14:40:11Z
sg-0a5fce4b0bbfa5621	launch-wizard-3	launch-wizard-3 created 2020-08-05T14:40:11Z

The screenshot shows the AWS Security Groups Inbound rules configuration page. The top navigation bar includes tabs for Inbound rules and Info. Below the tabs, there are filters for Type (SSH), Protocol (TCP), Port range (22), and Source (Info). The Source dropdown menu is open, showing "Custom", "Custom", "Anywhere", and "My IP", with "My IP" highlighted by a red circle. An "Add rule" button is located below the source dropdown.

- ❖ Create a new security group if you don't have one
- ❖ Allow traffic only from your IP address
- ❖ Don't create a new security group every time you launch a new instance
- ❖ Don't allow all traffic
- ❖ Name your security group starting with your name, e.g. 'jana\_security\_group'

# Launching an instance — Step 7

The screenshot shows the AWS Launch Wizard Step 7: Review Instance Launch page. At the top, there are tabs for 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review. The 7. Review tab is selected. Below the tabs, a section titled "Step 7: Review Instance Launch" contains a message: "Please review your instance launch details. You can go back to edit changes for each section. Click Launch to assign a key pair to your instance and complete the launch process." A warning message in a yellow box says: "⚠ Improve your instances' security. Your security group, launch-wizard-1, is open to the world. Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only. You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)". The "AMI Details" section shows an Ubuntu Server 18.04 LTS (HVM) AMI with SSD Volume Type. The "Instance Type" section shows a t2.micro instance type with 1 vCPU, 1 GiB memory, EBS only storage, and low to moderate network performance. The "Security Groups" section shows a single security group named "launch-wizard-1" created on 2020-04-03T19:24:25.538-04:00. The "Inbound Rules" table shows an SSH rule on port 22 from 0.0.0.0/0 using TCP protocol. The "Instance Details" section is partially visible at the bottom.

The screenshot shows a dialog box titled "Select an existing key pair or create a new key pair". It contains instructions: "A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance." Below this, a note says: "Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#)." The "Choose an existing key pair" dropdown has "jana\_key" selected. A checkbox below it is checked with the text: "I acknowledge that I have access to the selected private key file (jana\_key.pem), and that without this file, I won't be able to log into my instance." At the bottom are "Cancel" and "Launch Instances" buttons.

❖ Review your settings and launch your instance using your AWS key

# CELL RANGER



COLUMBIA UNIVERSITY  
IRVING MEDICAL CENTER

# SSH into your instance

The screenshot shows the AWS CloudWatch Instances dashboard. At the top, there's a table titled "Instances (1/7) Info" with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm Status, and Available. An arrow points to the checkbox next to the instance named "yiping\_instance\_large". Below this, a modal window is open for the instance "i-0111cac05d48d23dd (yiping\_instance\_large)". The modal has tabs for Details, Security, Networking, Storage, Status Checks, Monitoring, and Tags. The Details tab is selected. Under "Instance summary", the instance ID is listed as "i-0111cac05d48d23dd (yiping\_instance\_large)" and the state is "Running". The Public IPv4 address is "3.137.179.22" and the Public IPv4 DNS is "ec2-5-137-179-22.us-east-2.compute.amazonaws.com". A blue oval highlights the Public IPv4 DNS field.

- ssh (Secure Shell) is used to securely connect to a remote server/system (it transfers the data in encrypted form between the host and the client)
- Select your instance and copy the IP address ([Public IPv4 DNS](#))
- Mount the remote folder to your local folder by specifying the [directory of your AWS key](#) and the IP address of the instance

```
# Create local folder as mounting point
```

```
mkdir AWS
```

```
# Mount your instance's home directory to your local AWS folder
```

```
sshfs -o IdentityFile=XXX.pem ubuntu@ec2-3-137-186-247.us-east-2.compute.amazonaws.com:/home/ubuntu/ ~/AWS
```

```
# SSH into your instance by specifying the directory of your AWS key and the IP address of the instance
```

```
ssh -Y -i XXX.pem ubuntu@ec2-3-137-186-247.us-east-2.compute.amazonaws.com
```

# Update instance and install AWS CLI

- Update instance with:

```
sudo apt-get update; sudo apt-get upgrade -y
```

- Connect to your AWS account:

```
# install aws command line
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install

# set up aws s3 access
mkdir ~/.aws
echo "[default]" >> .aws/credentials
echo "aws_access_key_id = XXX" >> .aws/credentials
echo "aws_secret_access_key = XXX" >> .aws/credentials

# list all buckets
aws s3 ls
```

# Download and install Cell Ranger

- Copy download link for latest Cell Ranger version from: <https://support.10xgenomics.com/single-cell-gene-expression/software/downloads/latest>

```
curl -o cellranger-4.0.0.tar.gz ...
```

```
tar -zxvf cellranger-4.0.0.tar.gz
```

```
export PATH=/home/ubuntu/cellranger-4.0.0:$PATH
```

- Check if Cell Ranger works

```
cellranger
```

- Copy download link for latest ‘Human reference (GRCh38) dataset required for Cell Ranger’

```
curl ...
```

```
mkdir reference
```

```
tar -zxvf refdata-gex-GRCh38-2020-A.tar.gz -C reference/
```

```
rm *.tar.gz
```

**UPDATE:**  
In the latest versions you  
don't need to generate a  
custom reference anymore  
for snRNA-seq (-> skip this)

# Create a "pre-mRNA" reference for snRNA-seq

## 1. Create a "pre-mRNA" GTF

Extract GTF annotation rows for transcripts based on the feature type transcript (column 3) of the original tab-delimited GTF and replace the feature type from transcript to exon.

```
# Examine GTF file structure
head reference/refdata-gex-GRCh38-2020-A/genes/genes.gtf

# Check gene transcript locus labels in original file
awk '{print $3}' reference/refdata-gex-GRCh38-2020-A/genes/genes.gtf | sort | uniq -c

# List each gene transcript locus as an exon
awk 'BEGIN{FS="\t"; OFS="\t"} $3 == "transcript"{ $3="exon"; print}' reference/refdata-gex-GRCh38-2020-A/genes/genes.gtf > references/GRCh38-2020-A_premrna.gtf

# Check gene transcript locus labels in new file
awk '{print $3}' reference/GRCh38-2020-A_premrna.gtf | sort | uniq -c
```

# Create a "pre-mRNA" reference for snRNA-seq

**UPDATE:**  
In the latest versions you  
don't need to generate a  
custom reference anymore  
for snRNA-seq (-> skip this)

## 2. Run cellranger mkref

Use the unmodified genome.fa file and the new GTF file as inputs to cellranger mkref.

```
cd reference
nohup cellranger mkref --genome=GRCh38-2020-A_premrna \
    --fasta=refdata-gex-GRCh38-2020-A/fasta/genome.fa \
    --genes=GRCh38-2020-A_premrna.gtf \
    --nthreads=16 > ref.out 2> ref.err &

# Check if mkref is running (leave with control + c)
cd
htop

# Press F9 to kill a job in htop
```

# Transfer reference from S3 to instance

**UPDATE:**  
In the latest versions you  
don't need to generate a  
custom reference anymore  
for snRNA-seq (-> skip this)  
**Use regular cellranger ref  
instead with introns flag**

```
# transfer between instance and S3
cd ~
aws s3 sync s3://workshop-aws-data/reference/ reference/ --exclude '*' --include 'premrna' --dryrun
```



Source: S3 storage      Target: your instance      Exclude everything apart from files with 'premrna'

- Directories have to end with /
- Use --dryrun first to make sure you selected the right directories
- Rerun without --dryrun

# Create a volume

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with various services like EC2 Dashboard, Events, Tags, Instances, Images, and Elastic Block Store. Under 'Elastic Block Store', the 'Volumes' option is circled with a red oval. At the top right, there's a 'Create Volume' button with a red arrow pointing to it. Below the button is a search bar and a table listing volumes with columns for Name and Volume ID. A message at the bottom says 'Select a volume above'.

Volumes > Create Volume

## Create Volume

Volume Type: General Purpose SSD (gp2)

Size (GiB): 1 (Min: 1 GiB, Max: 16384 GiB)

IOPS: 100 / 3000 (Baseline of 3 IOPS per GiB with a minimum of 100 IOPS, burstable to 3000 IOPS)

Availability Zone\*: us-east-2a

Throughput (MB/s): Not applicable

Snapshot ID: Select a snapshot

Encryption:  Encrypt this volume

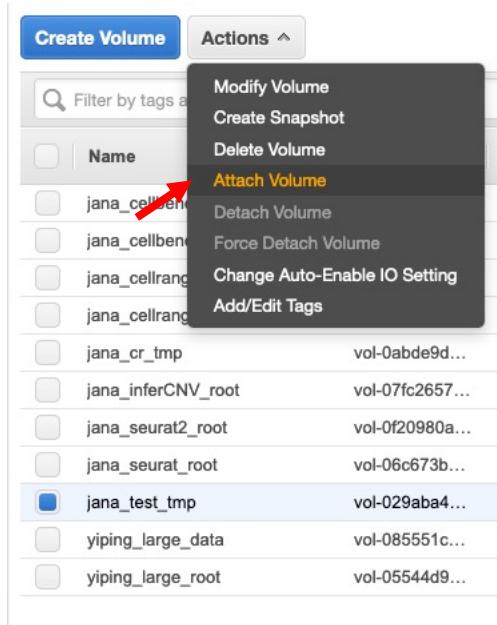
Tags:

Key	(128 characters maximum)	Value	(256 characters maximum)
Name		jana_test_tmp	

Add Tag 49 remaining (Up to 50 tags maximum)

- ❖ Name your instances and volumes starting with your name
- ❖ E.g. 'yiping\_cellranger\_tmp'
- ❖ Make sure volume is in same AZ as instance
- ❖ Today you'll need 200 GiB

# Attach and mount the volume to your instance



```
# Find the name of the volume (can be identified by size; it's probably nvme1n1)
lsblk
mkdir vol

# Format the volume to ext4 filesystem
sudo mkfs -t ext4 /dev/nvme1n1

# Mount volume
sudo mount /dev/nvme1n1 vol

# Change owner
sudo chown ubuntu:ubuntu vol
```

# Downloading raw data to instance

Today we'll transfer the raw data of patient 5 (melanoma brain metastasis) from S3

```
aws s3 sync s3://workshop-aws-data/raw_data/BI5/ vol/  
sudo chown ubuntu:ubuntu vol/*
```

Transfer the Cell Ranger scripts to your instance

```
aws s3 sync s3://workshop-aws-data/code/ ./ --exclude '*' --include 'cellranger*'
```

# Downloading data from Genewiz to instance

Usually you'll have to download the raw data from Genewiz

```
# connect to Genewiz remote server (have password ready)
sftp cumc_izar_orders_gmail@sftp.genewiz.com

# go to local destination folder (here vol)
lcd ~/vol

# navigate to the remote folder with your data using ls and cd

# download all files in remote directory to local directory
mget *

# usually you'll receive three fastq files: R1, R2, I1
```

# Run Cell Ranger: The main script

```
##### cellranger.sh #####
#!/bin/sh -x
# provide 'BI5'
PAT=$1
cd vol
date -u > time_${PAT}.log
# execute
cellranger count --id=${PAT} --localcores=16 --localmem=128 \
    --transcriptome=/home/ubuntu/reference/GRCh38-2020-A_premrna \
    --fastqs=/home/ubuntu/vol/ --sample=${PAT} --chemistry SC3Pv3
date -u >> time_${PAT}.log
cd ..
# sync out
aws s3 sync /home/ubuntu/vol/${PAT}/ s3://workshop-aws-data/your_name/cellranger/${PAT}/ --exclude "*.fastq.gz" --quiet
#####
```

Shebang: program loader is instructed to run the program /bin/sh, passing path/to/script as the first argument → tells computer how to read the script

Patient name will be passed as first variable, see next slide

Logging the time that Cell Ranger takes

Path to the snRNA-seq reference

Path to downloaded fastqs

Sync files out to S3 when done

threeprime for Single Cell 3',  
fiveprime for Single Cell 5',  
**SC3Pv3** for Single Cell 3' v3,  
SC5P-PE for Single Cell 5' paired-end (both R1 and R2 are used for alignment),  
SC5P-R2 for Single Cell 5' R2-only (where only R2 is used for alignment)

# Run Cell Ranger: The wrapper script

- The wrapper script calls the main script in a loop for each sample (makes more sense with multiple samples: `for i in BI1 BI2 BI3 BI4 BI5`)
- `nohup` and `&` make sure the script keeps running when you exit the instance
- The output and error files for each sample can be found in individual files in the home directory

```
##### cellranger_wrapper.sh #####
#!/bin/sh

cd ~

for i in BI5; do
    nohup cellranger.sh $i > cr_$i.out 2> cr_$i.err &
done
#####
```

# Run Cell Ranger

- Make the scripts executable using chmod

```
chmod +x cellranger.sh  
chmod +x cellranger_wrapper.sh
```

- Execute

```
./cellranger_wrapper.sh
```

- Check if it's working

```
htop
```

# Set an alarm for your instance — Create alarm and notifications

The screenshot shows the AWS CloudWatch Metrics and Alarms interface. On the left, there's a sidebar with 'Status check' and 'Alarm Status' sections. The 'Alarm Status' section is circled in red, and a red arrow points from it to the 'Create a new alarm' button in the main panel. The main panel has two tabs: 'Add or edit alarm' (selected) and 'Edit an existing alarm'. Under 'Add or edit alarm', there are two options: 'Create a new alarm' (selected) and 'Edit an existing alarm'. A search bar below these tabs says 'Select an existing alarm to edit'. In the 'Alarm notification' section, a dropdown menu is open, showing a search result 'jana\_new\_alarm' and a suggestion 'Create a new SNS Topic'. A red arrow points from the 'Create a new SNS Topic' suggestion to the suggestion itself.

- ❖ Always set alarms for your instances to shut down automatically when they are not used anymore
- ❖ Forgetting to shut down an instance can become expensive
- ❖ You can also create alarms for your volumes

# Set an alarm for your instance — Alarm action and threshold

The screenshot shows the configuration of an alarm for an Amazon instance. It consists of four panels:

- Alarm notification**: Info. Configure the alarm to send notifications to an Amazon SNS topic when it is triggered. A search bar contains "jana\_new\_alarm".
- Alarm action**: Info. Specify the action to take when the alarm is triggered. A dropdown menu shows "Selection action to alarm fires" with options: Recover, Reboot, Stop, and Terminate. The "Stop" option is highlighted with a red arrow.
- Alarm action**: Info. Specify the action to take when the alarm is triggered. A dropdown menu shows "Stop" selected.
- Alarm thresholds**: Info. Specify the metric thresholds for the alarm. Settings include:
  - Group samples by: Average
  - Type of data to sample: CPU Utilization
  - Alarm When: < 0.5 Percent
  - Consecutive Periods: 2
  - Period: 15 Minutes
  - Alarm name: jana\_new\_alarm

# Set an alarm for your instance — Define details

The screenshot illustrates the process of setting up a new alarm for an Amazon EC2 instance. It consists of two main parts:

- Top Panel (Modal):** A modal window titled "Alarm details for i-0bdd236fa1f93c94f" displays the name "jana\_new\_alarm" with a blue link icon. A red arrow points to this link. A "Close" button is at the bottom right.
- Bottom Panel (Main View):** The main CloudWatch Alarms page shows the newly created "jana\_new\_alarm".
  - Left Sidebar:** Shows a list of alarms: "jana\_new\_alarm" (selected, highlighted in blue), "jana\_seurat\_alarm", and "jana\_callhander\_alarm". Each entry includes a status indicator (blue dot for jana\_new\_alarm, grey dot for others) and a "Metric alarm" label.
  - Right Main Area:** The title "jana\_new\_alarm" is followed by a "Graph" section. The graph displays "CPUUtilization" over time from 06:00 to 17:00. A horizontal red line at 0.5 represents the threshold. The text "CPUUtilization < 0.5 for 2 datapoints within 30 minutes" is displayed above the graph. The legend indicates "CPUUtilization".
  - Actions Menu:** A context menu is open on the right, with "Edit" highlighted and a red arrow pointing to it. Other options include "Delete", "Copy", and "Add to dashboard".

# Set an alarm for your instance — Create SNS topic

CloudWatch > Alarms > jana\_new\_alarm > Edit

Step 1  
Specify metric and conditions

Step 2  
**Configure actions** →

Step 3  
Add name and description

Step 4  
Preview and create

### Configure actions

#### Notification

Alarm state trigger  
Define the alarm state that will trigger this action.

In alarm  
The metric or expression is outside the defined threshold.

OK  
The metric or expression is within the defined threshold.

Select an SNS topic  
Define the SNS (Simple Notification Service) topic that will receive the notification.

Select an existing SNS topic

Create new topic →

Use topic ARN

Create a new topic...  
The topic name must be unique.  
jana\_new\_alarm

SNS topic names can contain only alphanumeric characters, hyphens (-) and underscores (\_).

Email endpoints that will receive the notification...  
Add a comma-separated list of email addresses. Each address will be added:  
jb4424@cumc.columbia.edu jb4424@cumc.columbia.edu

user1@example.com, user2@example.com

**Create topic** →

**Add notification**

jana\_new\_alarm > Edit

### Configure actions

#### Notification

Alarm state trigger  
Define the alarm state that will trigger this action.

In alarm  
The metric or expression is outside the defined threshold.

OK  
The metric or expression is within the defined threshold.

Insufficient data  
The alarm has just started or not enough data is available.

**Remove**

Select an SNS topic  
Define the SNS (Simple Notification Service) topic that will receive the notification.

Select an existing SNS topic

Create new topic

Use topic ARN

Send a notification to...  
jana\_new\_alarm jana\_new\_alarm

Only email lists for this account are available.

Email (endpoints)  
jb4424@cumc.columbia.edu - View in SNS Console ↗

**Add notification** →

Email:

[EXTERNAL] AWS Notification - Subscription Confirmation



AWS Notifications <no-reply@sns.amazonaws.com>  
To: Biermann, Jana

You have chosen to subscribe to the topic:  
arn:aws:sns:us-east-2:268169496675:jana\_new\_alarm

To confirm this subscription, click or visit the link below (if this was in error no action is necessary):  
[Confirm subscription](#) →

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation emails, click here to unsubscribe.



Simple Notification Service

**Subscription confirmed!**

You have subscribed jb4424@cumc.columbia.edu to the topic:  
jana\_new\_alarm.

Your subscription's id is:  
arn:aws:sns:us-east-2:268169496675:jana\_new\_alarm:f34be660-706c-4f75-959e-75c0a1d4232a

If it was not your intention to subscribe, [click here to unsubscribe](#).

# Set an alarm for your instance — Define EC2 action

**Auto Scaling action**

[Add Auto Scaling action](#)

**EC2 action**

Alarm state trigger  
Define the alarm state that will trigger this action.

In alarm  
The metric or expression is outside of the defined threshold.

OK  
The metric or expression is within the defined threshold.

Insufficient data  
The alarm has just started or not enough data is available.

[Remove](#)

Take the following action...

Define what will happen to the EC2 instance with the Instance ID i-0bdd236fa1f93c94f when this alarm is triggered.

Recover this instance  
You can only recover certain EC2 instance types. [See documentation](#)

Stop this instance  
You can only stop an instance if it is backed by an EBS volume. AWS will use the existing Service Linked Role (AWSServiceRoleForCloudWatchEvents) to perform this action. [Show IAM policy document](#)

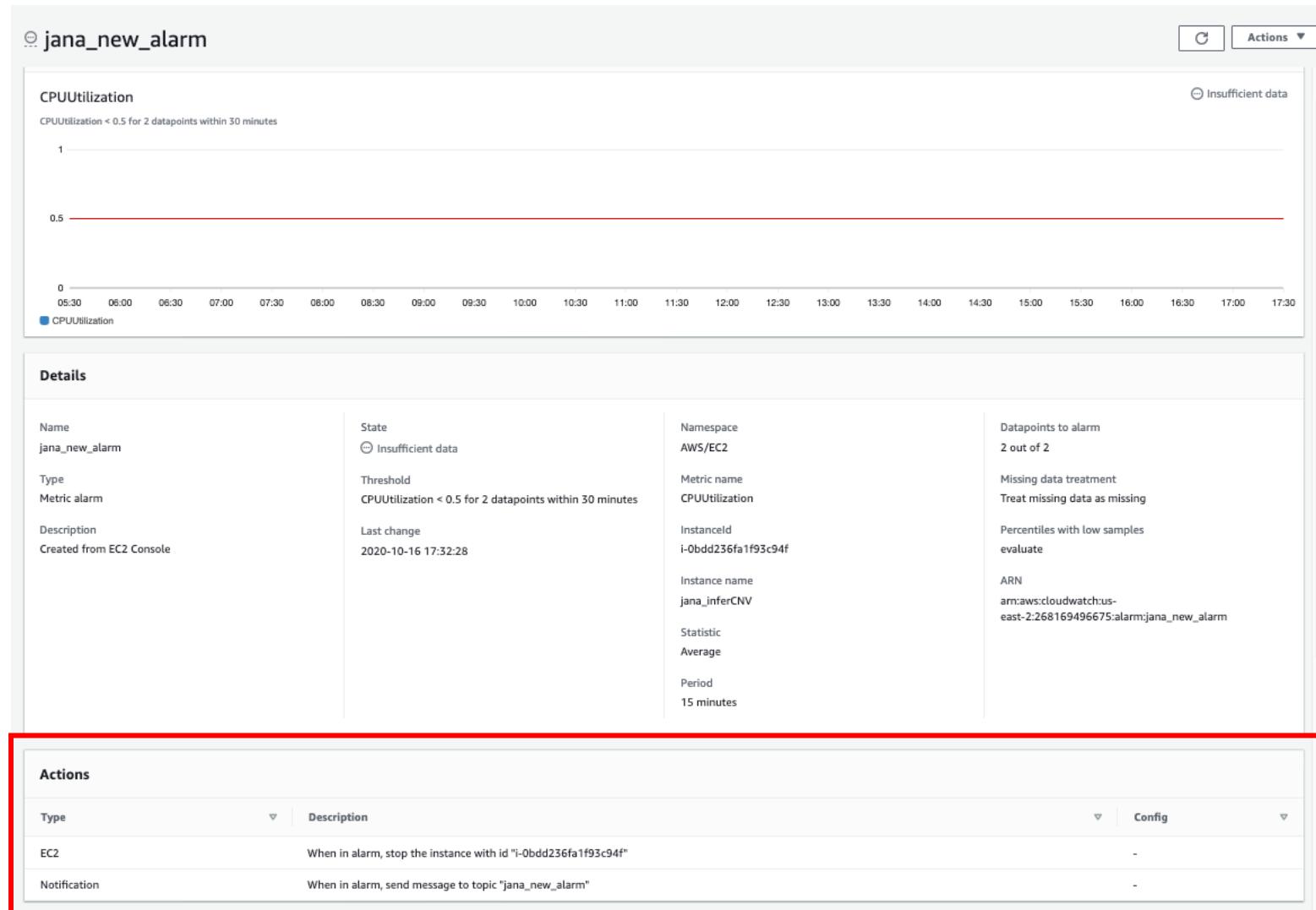
Terminate this instance  
You will not be able to terminate this instance if termination protection is enabled. AWS will use the existing Service Linked Role (AWSServiceRoleForCloudWatchEvents) to perform this action. [Show IAM policy document](#)

Reboot this instance  
An instance reboot is equivalent to an operating system reboot. AWS will use the existing Service Linked Role (AWSServiceRoleForCloudWatchEvents) to perform this action. [Show IAM policy document](#)

[Add EC2 action](#)

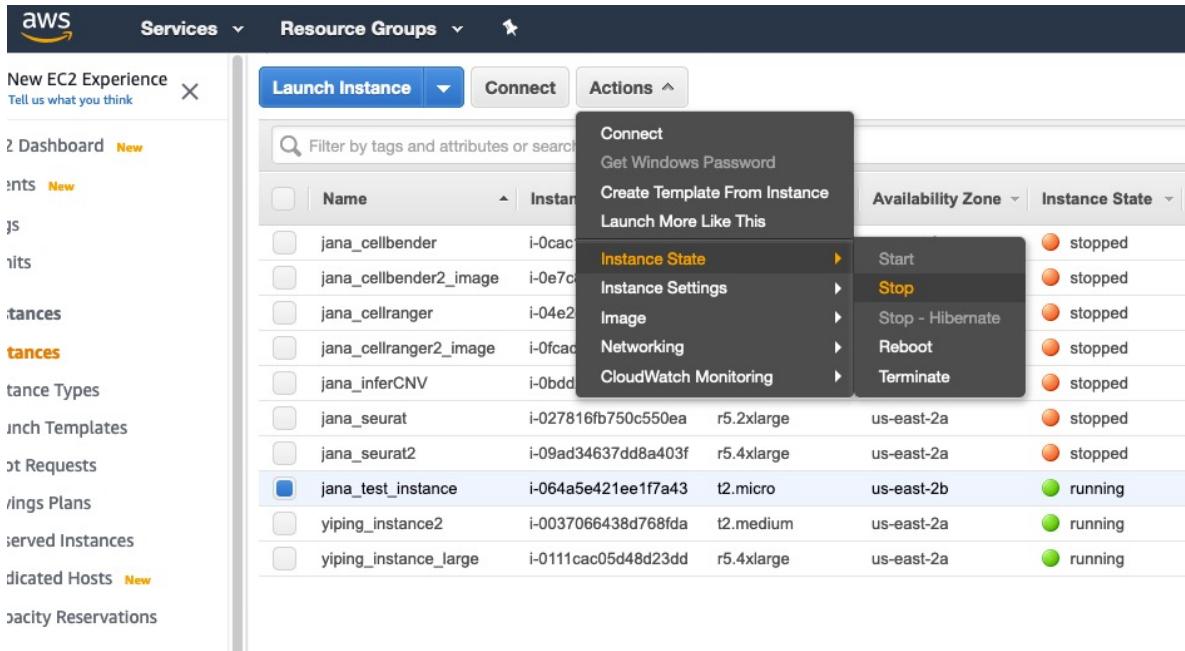
[Cancel](#) [Previous](#) [Next](#) [Update alarm](#)

# Set an alarm for your instance — Check notifications and actions



# Tidying up

- Stop Cell Ranger on your instance using htop and F9
- Terminate instance
- Detach volume
- Delete volume



- ❖ Make sure only your instance is selected
- ❖ ‘Stop’ will shut down the instance until you start it again
- ❖ ‘Terminate’ will delete the instance (and the associated root volume unless you changed the settings)
- ❖ When instances are stopped, we only pay for the storage of their volume

# Cell Ranger output

s3://workshop-aws-data/example/cellranger/B15/outs

<input type="checkbox"/> Name ▾
<input type="checkbox"/> analysis
<input type="checkbox"/> filtered_feature_bc_matrix
<input type="checkbox"/> raw_feature_bc_matrix
<input type="checkbox"/> cloupe.clope
<input type="checkbox"/> filtered_feature_bc_matrix.h5
<input type="checkbox"/> metrics_summary.csv
<input type="checkbox"/> molecule_info.h5
<input type="checkbox"/> possorted_genome_bam.bam
<input type="checkbox"/> possorted_genome_bam.bam.bai
<input type="checkbox"/> raw_feature_bc_matrix.h5
<input type="checkbox"/> web_summary.html

Can be used as Seurat input if you decide not to run CellBender first

Unfiltered version; can also be used as Seurat input in case you feel Cell Ranger filtered too strictly

Input for CellBender

Gives you a quick overview of your settings, QC metrics and t-SNE

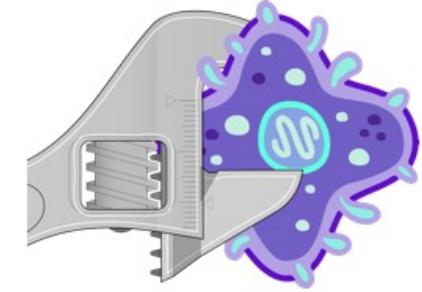
# CELLBENDER



COLUMBIA UNIVERSITY  
IRVING MEDICAL CENTER

# CellBender

- GitHub: <https://github.com/broadinstitute/CellBender>
- Preprint: <https://www.biorxiv.org/content/10.1101/791699v1>
- Model learns background RNA profile
- Distinguishes cell-containing droplets from empty droplets
- Generates background-free gene expression profiles
- CellBender requires access to Graphics Processing Units (GPUs)
  - GPU instance is recommended for most deep learning purposes
  - Training new models will be faster on a GPU instance than a CPU instance
  - Enable more parallelism for higher throughput on compute-intensive workloads
- CellBender uses CUDA (Compute Unified Device Architecture)
  - Parallel computing platform and application programming interface (API) model created by Nvidia
  - Allows to use a CUDA-enabled GPU for general purpose processing
  - CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements

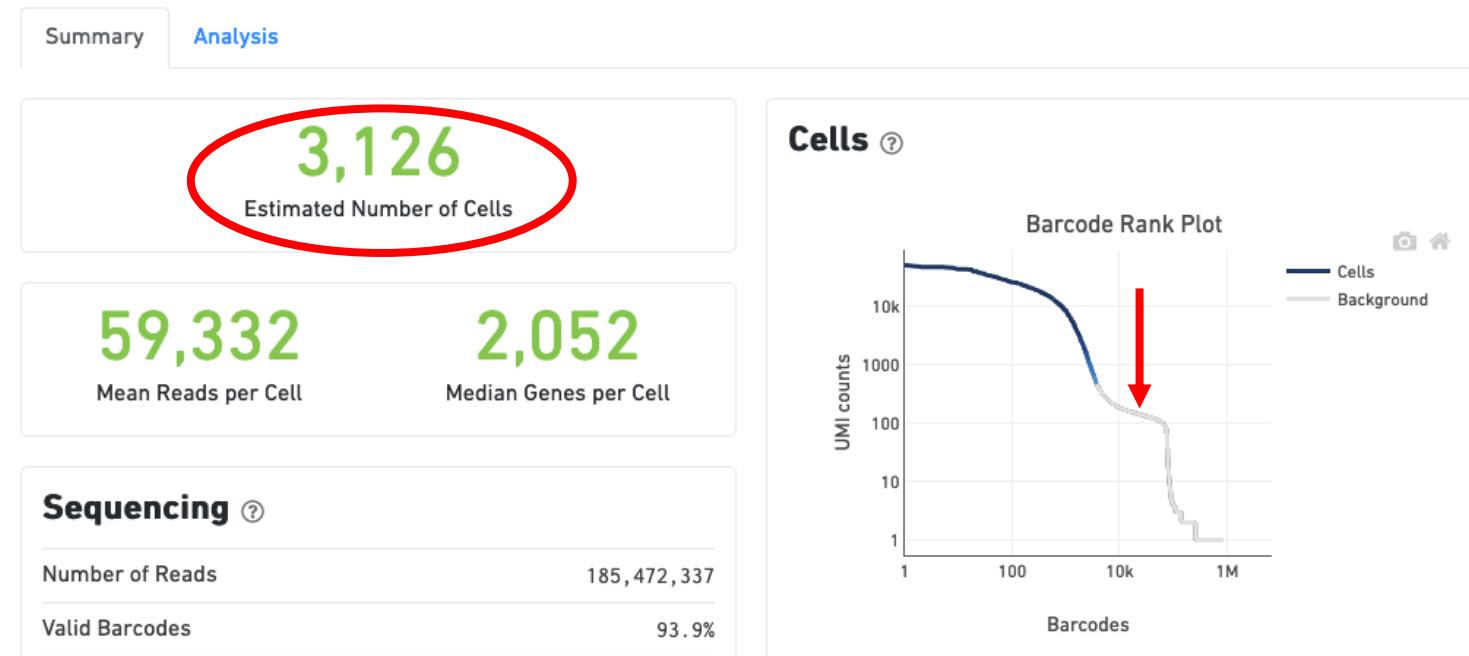


# Getting the parameters for CellBender

- Open web summary in S3:

[workshop-aws-data/example/cellranger/Bl5/outs/web\\_summary.html](workshop-aws-data/example/cellranger/Bl5/outs/web_summary.html)

- You will need to look up two parameters for each sample individually:
  - Estimated Number of Cells: ~3,100
  - Total droplets included: 12,000

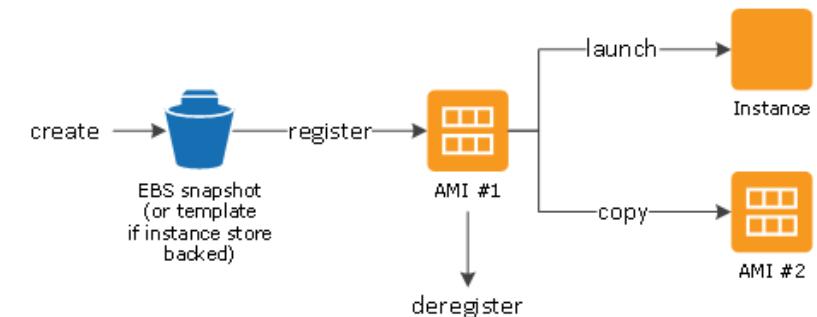


# Starting an AMI CellBender instance

The screenshot shows the AWS EC2 Dashboard with the 'AMIs' section selected. The 'Launch' button is highlighted with a red arrow. Another red arrow points to the selected AMI named 'foreveryone\_cellbender'. The table lists several AMIs:

Name	AMI Name	AMI ID
foreveryone_cellbender	foreveryone_cellbender	ami-04ea2b8e
foreveryone_cellranger4.0	foreveryone_cellranger4.0	ami-01966a6e
jana_cellbender	cellbender	ami-0884139c
jana_cellranger	cellranger_image	ami-0d72337a
jana_cellranger4.0	jana_cellranger4.0	ami-094081bc

- Amazon Machine Image (AMI):
  - Provides the information required to launch an instance
  - EBS snapshots serve as a template for the root volume of the instance (for example, an operating system, an application server, and applications)
- Choose instance p2.xlarge



# Setting up the CellBender instance

1. Update your instance

2. Add your AWS credentials

3. Verify that your instance has a GPU

```
lspci | grep -i nvidia
```

4. Activate the Conda environment

```
conda activate CellBender
```

5. Transfer the CellBender scripts to your instance

```
aws s3 sync s3://workshop-aws-data/code/ ./ --  
exclude '*' --include 'cellbender*'
```

# Run CellBender: The main script

```
##### cellbender.sh #####
#!/bin/sh
```

```
pat=$1
cells=$2
drop=$3
```

} Take the sample name, number of cells and droplet cut-off from the provided arguments (see next slide)

```
aws s3 sync s3://workshop-aws-data/example/cellranger/${pat}/outs/ data/${pat}/ --exclude '*' --include 'raw_feature_bc_matrix.h5'
sleep 15
```

} Transfer the raw matrix from S3

```
cellbender remove-background \
    --input data/${pat}/raw_feature_bc_matrix.h5 \
    --output data/${pat}/${pat}.h5 \
    --expected-cells ${cells} \
    --total-droplets-included ${drop} \
    --cuda \
    --epochs 300
```

} Obtain the number of cells and droplet cut-off from the provided arguments

```
aws s3 sync data/${pat}/ s3://workshop-aws-data/your_name/cellbender/${pat}/ --exclude "*.out" --exclude "*.err" --exclude 'raw_feature*' --exclude '.*' --quiet
#####

```

} Transfer the results to your S3 folder

# Run CellBender: The wrapper script

```
##### cellbender_wrapper.sh #####
#!/bin/sh

pat=$1
cells=$2
drop=$3

nohup ./cellbender.sh $pat $cells $drop > cb_$pat.out 2> cb_$pat.err &
#####

```

- Execute

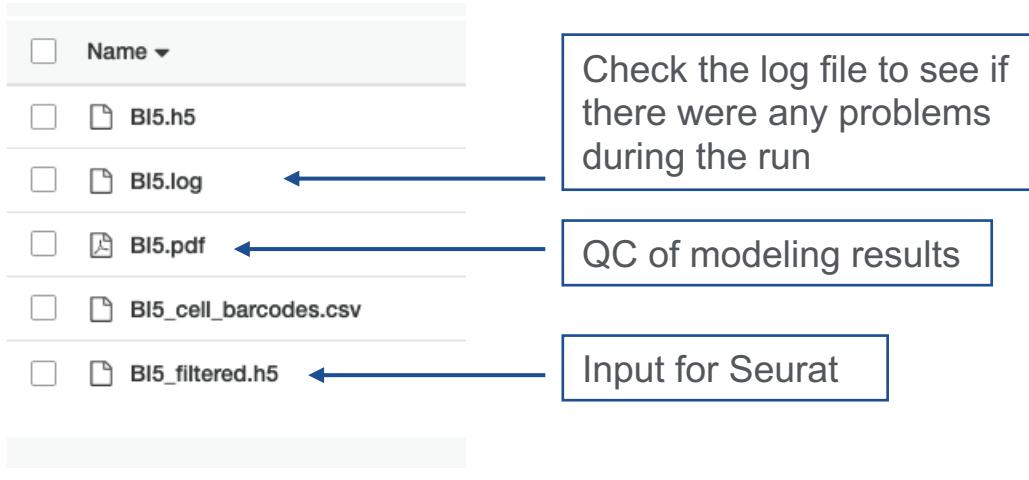
Provided arguments with the script:  
\$1 -> patient name  
\$2 -> expected-cells  
\$3 -> total-droplets-included

./cellbender\_wrapper.sh BI5 3100 12000

Execute this script located in the current directory

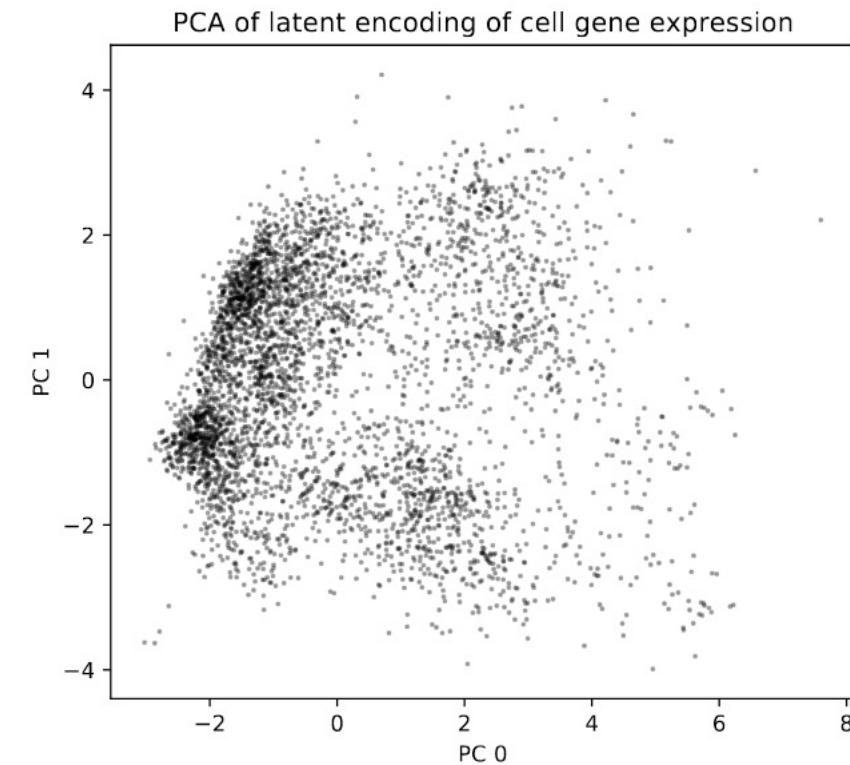
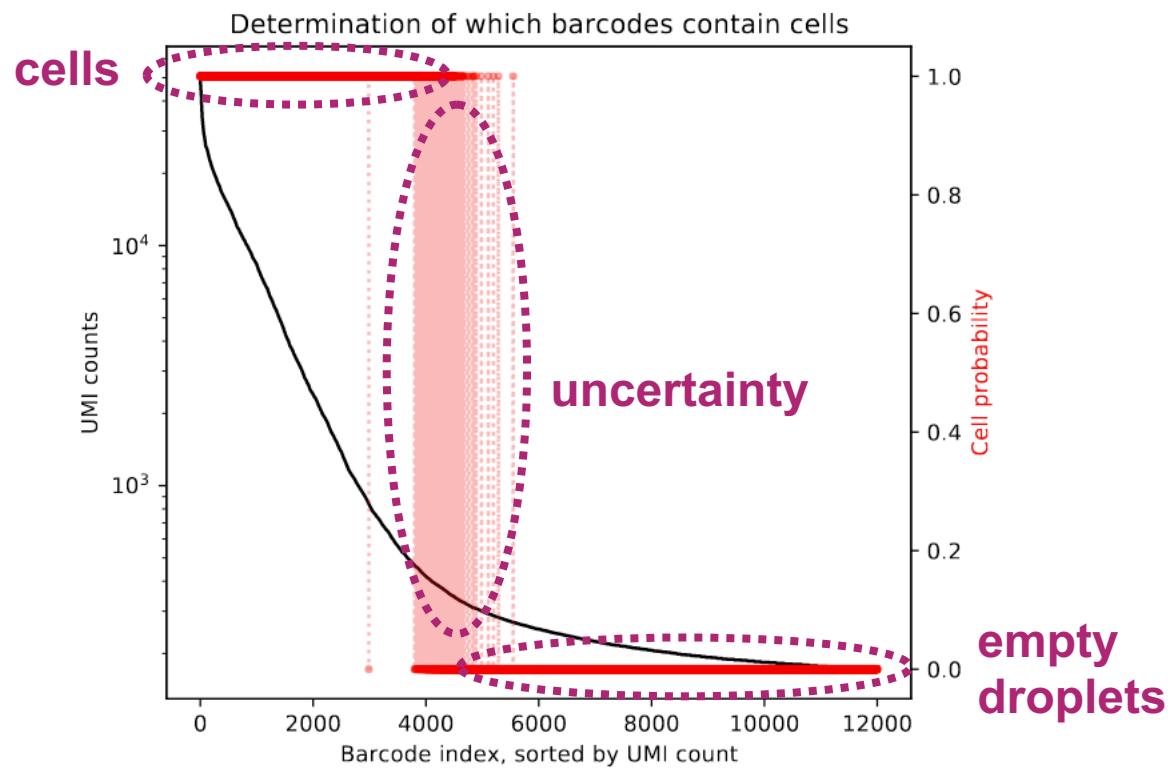
# CellBender output

s3://workshop-aws-data/example/cellbender/BI5/



- The first graph should be approaching a saturation threshold
- Otherwise increase number of epochs (will take longer)

# CellBender output



- The second plot shows the distribution of cells and empty droplets and the certainty of classification as either
- The third plot ideally shows several separate clusters (depends on what you sequenced)

# SEURAT



COLUMBIA UNIVERSITY  
IRVING MEDICAL CENTER

# Setting up Seurat

- We'll analyze the sample locally in Rstudio today
- Download the file BI5\_filtered.h5 from the CellBender output and save it locally in `~/AWS_workshop/data/BI5`
- Please name the folders exactly as suggested above
- Move the files seuratworkshop.R, scrublet\_code.py, and dummy\_signatures.csv into the folder `~/AWS_workshop/`
- Open seuratworkshop.R using Rstudio

# Loading single-cell data into Seurat

```
library(Seurat)
library(ggplot2)
library(Matrix)
library(SingleR)
library(SingleCellExperiment)
library(pheatmap)

#samplename is prefix of h5 file, for example BI5 in BI5_filtered.h5
#directoryname is name of folder where scripts, in this case ~/AWS_workshop
#h5 data file is then expected to be in the directory ~/AWS_workshop/data/BI5
#doubts is expected doublet rate, for later scrubbing of doublets
samplename <- "BI5"
directoryname <- "~/AWS_workshop"
doubts <- as.numeric(commandArgs()[3]) #.04 for this workshop

# Load dataset with Read10x_h5
#first argument is file name
#use.names = TRUE makes rows of the data be labeled with gene names, not gene ids
#unique.features = TRUE changes gene names in data, if necessary, to make them unique
seu.data <- Read10X_h5(paste0(directoryname,"/data/",samplename,"/",samplename,"_filtered.h5"), use.names = TRUE, unique.features = TRUE)

# Initialize the Seurat object with the raw (non-normalized data)
#project argument sets project name
#min.cells = 1, include genes that occur in at least one cell
#min.features = 1, include cells that have at least one gene
seu_raw <- CreateSeuratObject(counts = seu.data, project = samplename, min.cells = 1, min.features = 1)

#include patient name metadata
seu_raw[["patient"]]<-samplename
```

# Quality control filtering

```
#determine mitochondrial percentage
#mitochondrial gene names begin with MT-
#PercentageFeatureSet finds percentage of all genes that start with MT-
#store in new variable percent.mt
seu_raw[["percent.mt"]] <- PercentageFeatureSet(seu_raw, pattern = "^MT-")

# identify doublets using scrublet
#expected_doublet_rate of .04 is based on experimental data, accurate value helps to guide search for true doublets
#use writeMM to write out raw count data to a file
#use a python script to read in the count file, run scrublet to remove doublets, and write out the result to another file
#read in results from that file back into R
matrixfilename = paste0(directoryname,"/",samplename,"_MM.txt")
scrubletoutputfile = paste0(directoryname,"/",samplename,"_doublets.txt")
writeMM(seu_raw@assays$RNA@counts,file=matrixfilename)
system(paste0("python3 scrublet_code.py ",matrixfilename," ",doubs," ",scrubletoutputfile))
scrubletoutput = read.table(scrubletoutputfile,header=T,quote=NULL)
#store doublet status, and score for strength of doublet prediction, in two new variables in seu_raw
seu_raw[['ScrubDoublet']]<-scrubletoutput$predicted_doublets
seu_raw[['ScrubDoublet_score']]<-scrubletoutput$doublet_scores

#use subset function to filter data based on min and max number of features (genes),
# min and max number of reads (counts), maximum mitochondrial percentage, and removing all doublets
minFeature<-200
maxFeature<- 7500
minCount<- 100
maxCount<- 40000
maxMT<-10
seu <- subset(seu_raw, subset = nFeature_RNA > minFeature & nFeature_RNA < maxFeature & nCount_RNA > minCount
& nCount_RNA < maxCount & percent.mt < maxMT & ScrubDoublet ==F)
```

# Scrublet filtering with Python

```
#!/usr/bin/env python3

import sys
import scrublet as scr
import scipy.io
import sys

#sys.argv is a list of words that are given as arguments for Python script
#in this case, three inputs are the name of the input matrix file written by R,
#the expected doublet rate, and the name of the output file to write to
inputfile=sys.argv[1]
doublet_rate=float(sys.argv[2])
outputfile=sys.argv[3]

#read in matrix inputfile with imread function, then format it further with .T and tocsc()
counts_matrix = scipy.io.mmread(inputfile).T.tocsc()
#initialize Scrublet object, using matrix and expected doublet rate
scrub = scr.Scrublet(counts_matrix, expected_doublet_rate=doublet_rate)
#analyze genes with min RNA count number>=1 in >=1 cells
#analyze genes whose variability is at or above the 85th percentile, and take 20 components of the PCA of these variable genes
doublet_scores, predicted_doublets = scrub.scrub_doublets(min_counts=1, min_cells=1, min_gene_variability_pctl=50, n_prin_comps=20)

#write out doublet results to output file
outFl = open(outputfile,'w')
outFl.write('predicted_doublets\tdoublet_scores\n')
for i in range(len(predicted_doublets)):
    outFl.write(str(predicted_doublets[i]).upper()+'\t'+str(doublet_scores[i])+'\n')

outFl.close()
```

# Plot nCount vs. nFeature to check for quality control thresholds

```
pdf("workshop_QC_scatter_plot.pdf")
```

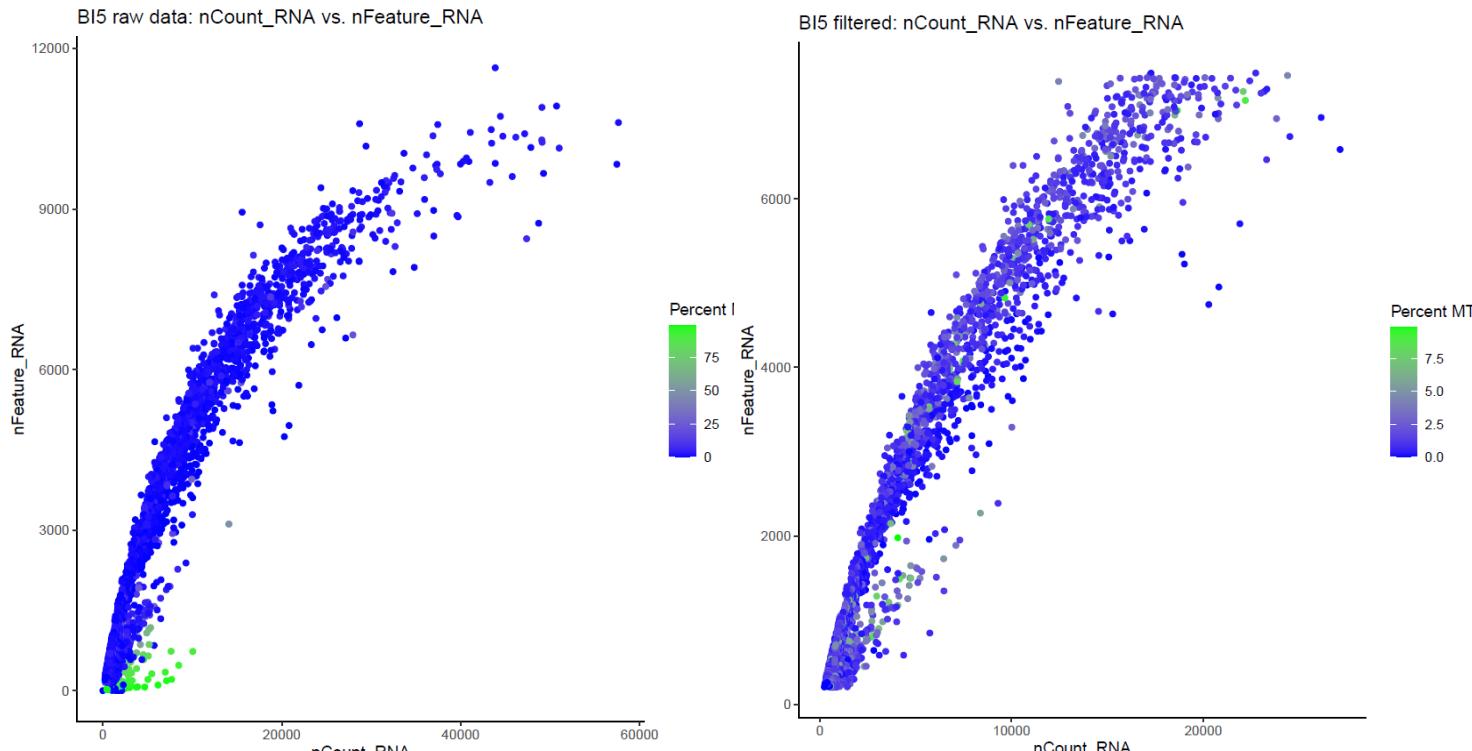
```
#print scatter plot for seu_raw using qplot function, with x axis being  
nCount, y nFeature, and color being percent.mt  
#xlab, ylab, and main set x axis, y axis, and overall figure title  
#scale_colour_gradient controls percent.mt colors  
#labs function controls legend label for color  
#theme_classic() changes overall graphic appearance  
print(qplot(x=seu_raw@meta.data$nCount_RNA,y =  
seu_raw@meta.data$nFeature_RNA,  
col=seu_raw@meta.data$percent.mt, xlab = "nCount_RNA",ylab =  
"nFeature_RNA", main =paste0(seu_raw@meta.data$orig.ident[1], "  
raw data: nCount_RNA vs. nFeature_RNA")) +  
scale_colour_gradient(low="blue", high="green") + labs(color =  
"Percent MT") + theme_classic()
```

```
#same plot for quality filtered object seu
```

```
print(qplot(x=seu@meta.data$nCount_RNA,y =  
seu@meta.data$nFeature_RNA, col=seu@meta.data$percent.mt,  
xlab = "nCount_RNA",  
ylab = "nFeature_RNA", main =paste0(seu@meta.data$orig.ident[1], "  
filtered: nCount_RNA vs. nFeature_RNA")) +  
scale_colour_gradient(low="blue", high="green") + labs(color =  
'Percent MT') + theme_classic())
```

```
dev.off()
```

```
#filtering low feature/count cells removes technical noise  
#filtering high feature/count cells remove likely doublets
```



# Canonical workflow to normalize data, and run PCA and UMAP

```
# normal workflow
#Log-normalizes data
seu <- NormalizeData(seu)

#find 2000 most variable genes in data
seu = FindVariableFeatures(seu, selection.method = "vst", nfeatures = 2000)

#shifts mean expression across cells 0, variance across cells to 1
seu <- ScaleData(object = seu)

#run PCA, to map cells to 2D PCA coordinates
seu <- RunPCA(object = seu, npcs = 100)

#Using dimensions 1-15 of the pca reduction, find closest neighboring cells to each single cell in dataset
seu <- FindNeighbors(seu, reduction = "pca", dims = 1:15)

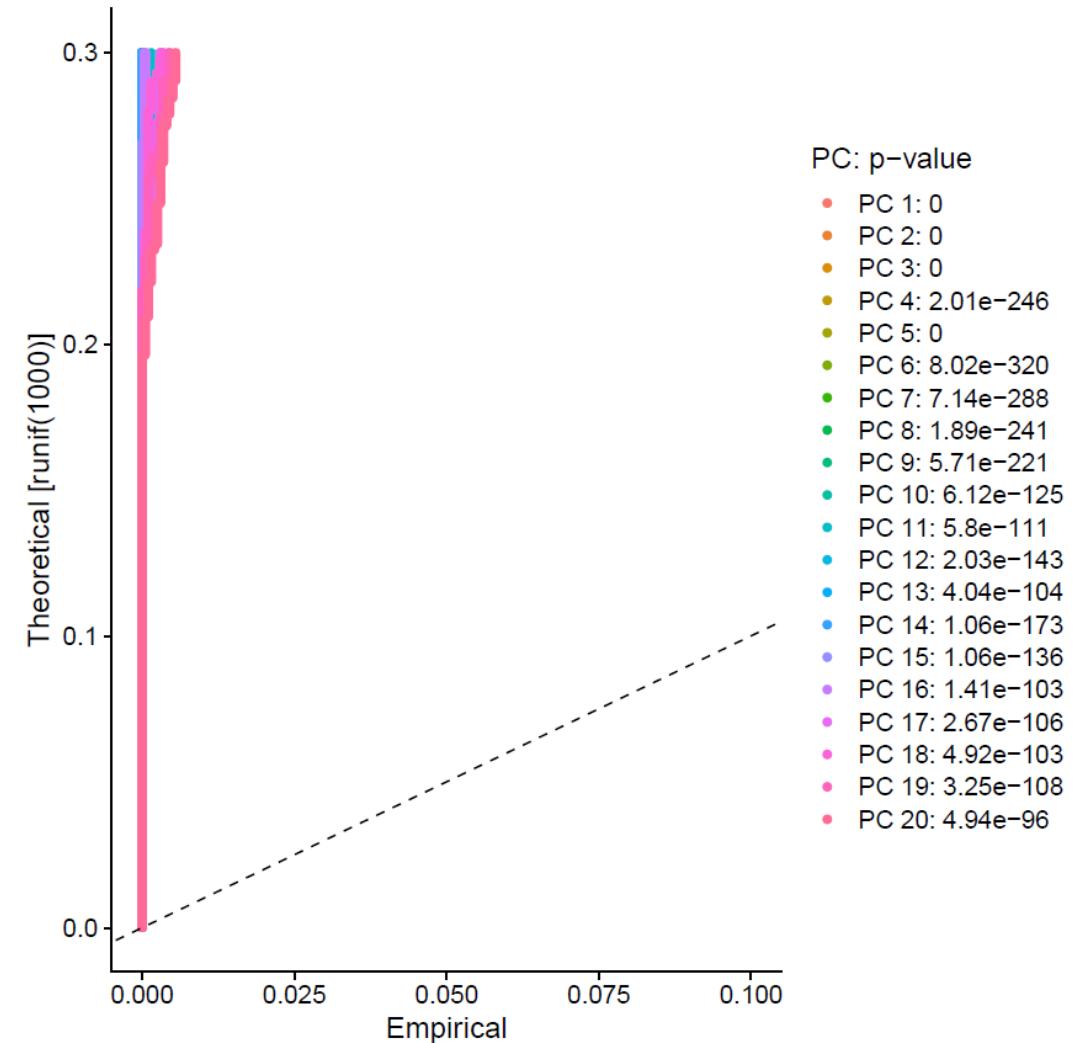
#determine number of clusters within dataset, based on neighbors found above
seu <- FindClusters(seu)

#run UMAP, to map cells to 2D UMAP coordinates, using dimensions 1-20 of the pca reduction
seu <- RunUMAP(object = seu, dims = 1:20)
```

# Determining number of significant PCs with Jackstraw

```
#Jackstraw procedure permutes expression of a subset of genes (default 1%)
#permute 100 times in num.replicate argument below
#rerun PCA, and add jackstraw scores for top 20 dimensions
seu <- JackStraw(seu, num.replicate = 100)
seu <- ScoreJackStraw(seu, reduction = "pca", dims = 1:20)
```

```
#For each PC, can then determine percentage of genes are associated with the
PC in permuted data, plotted on X axis
#On Y axis, plot percentage of genes associated with the PC in actual data
#Significant PCs are enriched in significantly associated genes, placed towards
left edge of graph
pdf("workshop_jackstraw_plot.pdf")
print(JackStrawPlot(seu, dims = 1:20))
dev.off()
```



# Print heatmap of genes on each end of each PCA component

```
#print PCA loadings for components 1 and 2  
pdf("workshop_pca_analysis.pdf")  
print(VizDimLoadings(seu, dims = 1:2,  
reduction = "pca"))
```

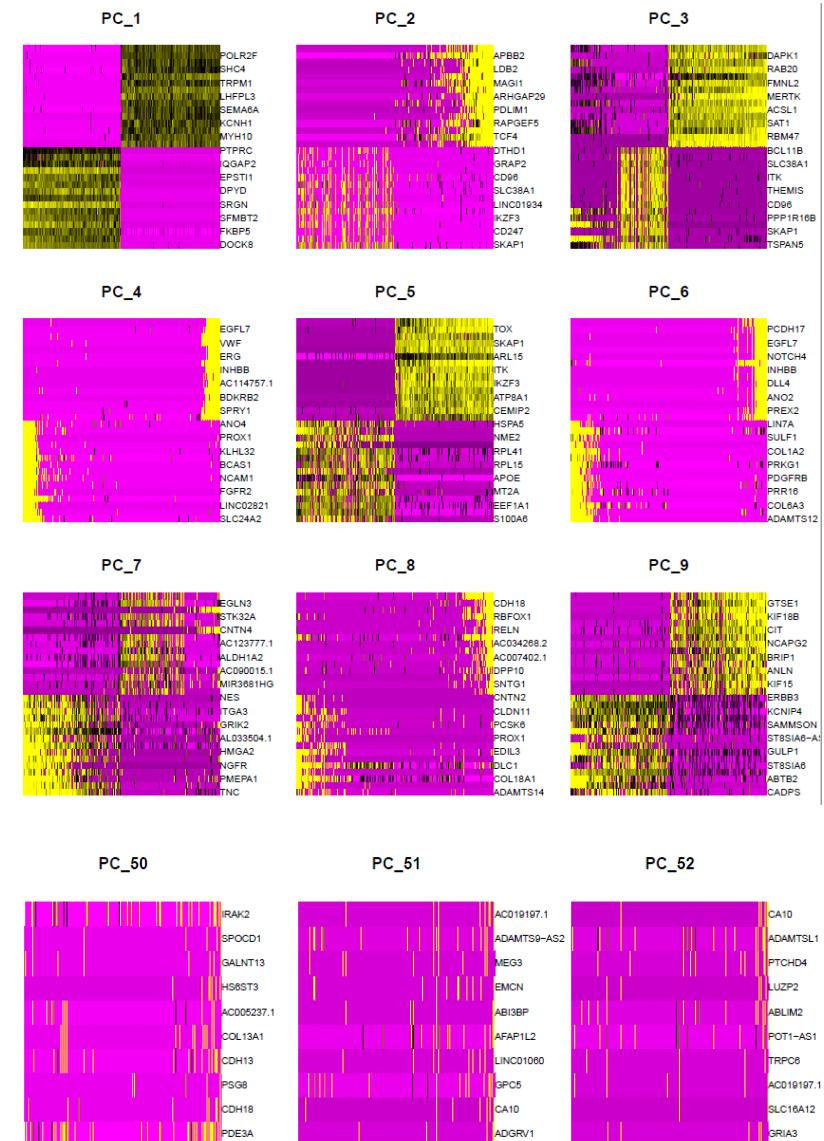
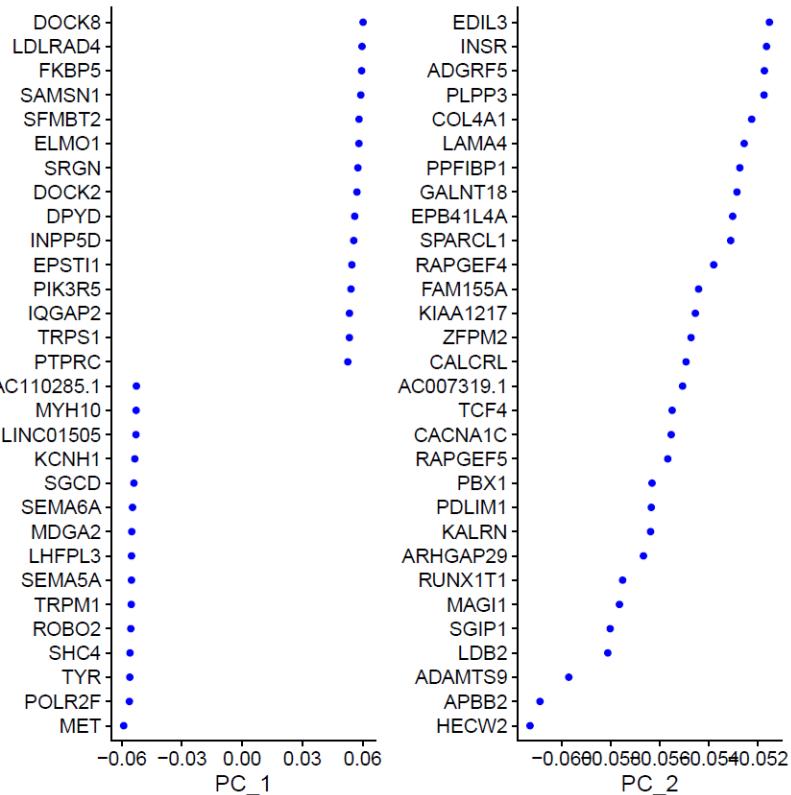
```
#cells are ordered along PCA component  
#print heatmap of gene expression for 500  
cells of seu, using balanced=TRUE, which  
selects half from the top of a PCA  
component, half from the bottom
```

#use PCA dimensions 1-9 for good PCs with structure, or 50-52 for bad ones without structure

```
print(DimHeatmap(seu, dims = 1:9, cells = 500, balanced = TRUE))
```

```
print(DimHeatmap(seu, dims = 50:52, cells =  
 500, balanced = TRUE))
```

`dev.off()`



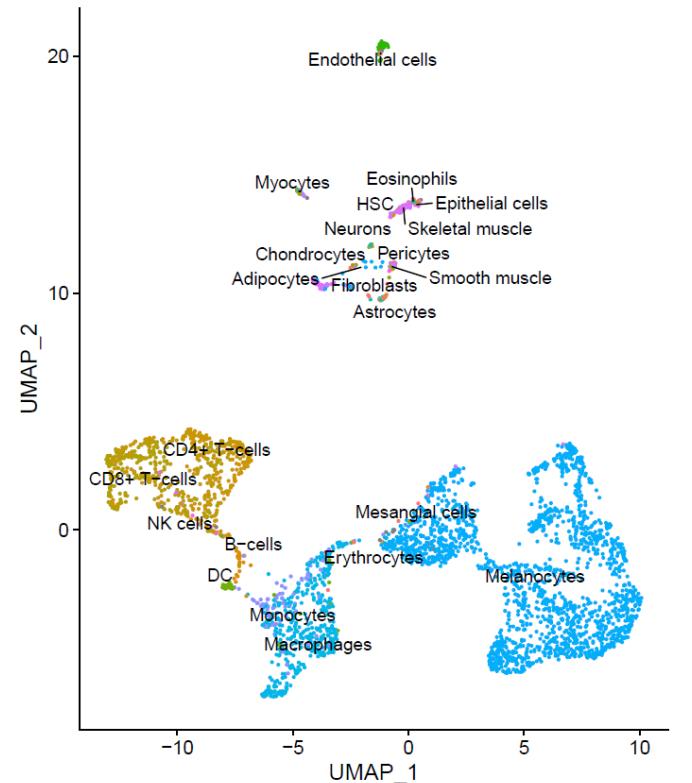
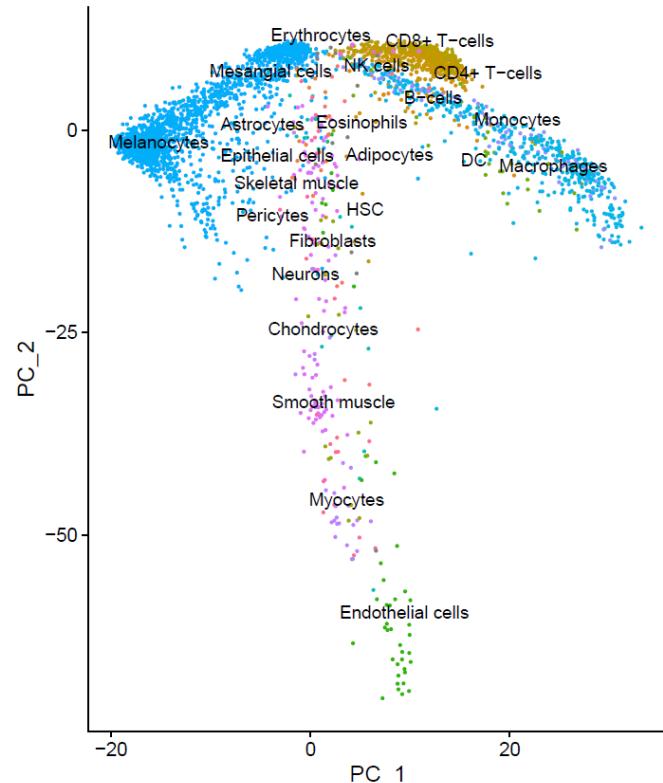
# Print PCA and UMAP visualizations of single cell data

```
#plot umaps and pcas in single file
#pdf function creates a new file called seu_integrated.pdf
pdf(file = "workshop_umap_and_pca.pdf")

#print function plots a new figure on each page of the pdf
#DimPlot plots a 2D PCA or UMAP reduction of seu
#points in PCA or UMAP are colored by subset given in
group.by argument
#label argument controls whether the groups are labeled
in the figure
#guides and theme function are to format the legend, so it
is a single column of labels, and takes less space in the
figure
print(DimPlot(seu, reduction = "pca",label = T, group.by =
'celltype_bp6_main') + guides(col = guide_legend(nrow =
30,override.aes = list(size=5))) +
theme(legend.text=element_text(size=6)))

print(DimPlot(seu, reduction = "umap",label = T,group.by =
'celltype_bp6_main') + guides(col = guide_legend(nrow =
30,override.aes = list(size=5))) +
theme(legend.text=element_text(size=6)))

dev.off()
```



# Find markers of differential expression

```
#find markers of differential expression between cluster 1 and all other cells  
#only.pos controls whether only positive DE markers are found, or negative  
as well  
#min.pct controls what percentage of cells must have non-zero expression of  
the marker  
#logfc.threshold controls minimum log-fc threshold  
cluster1.markers <- FindMarkers(seu, ident.1 = 1, min.pct = 0.25)

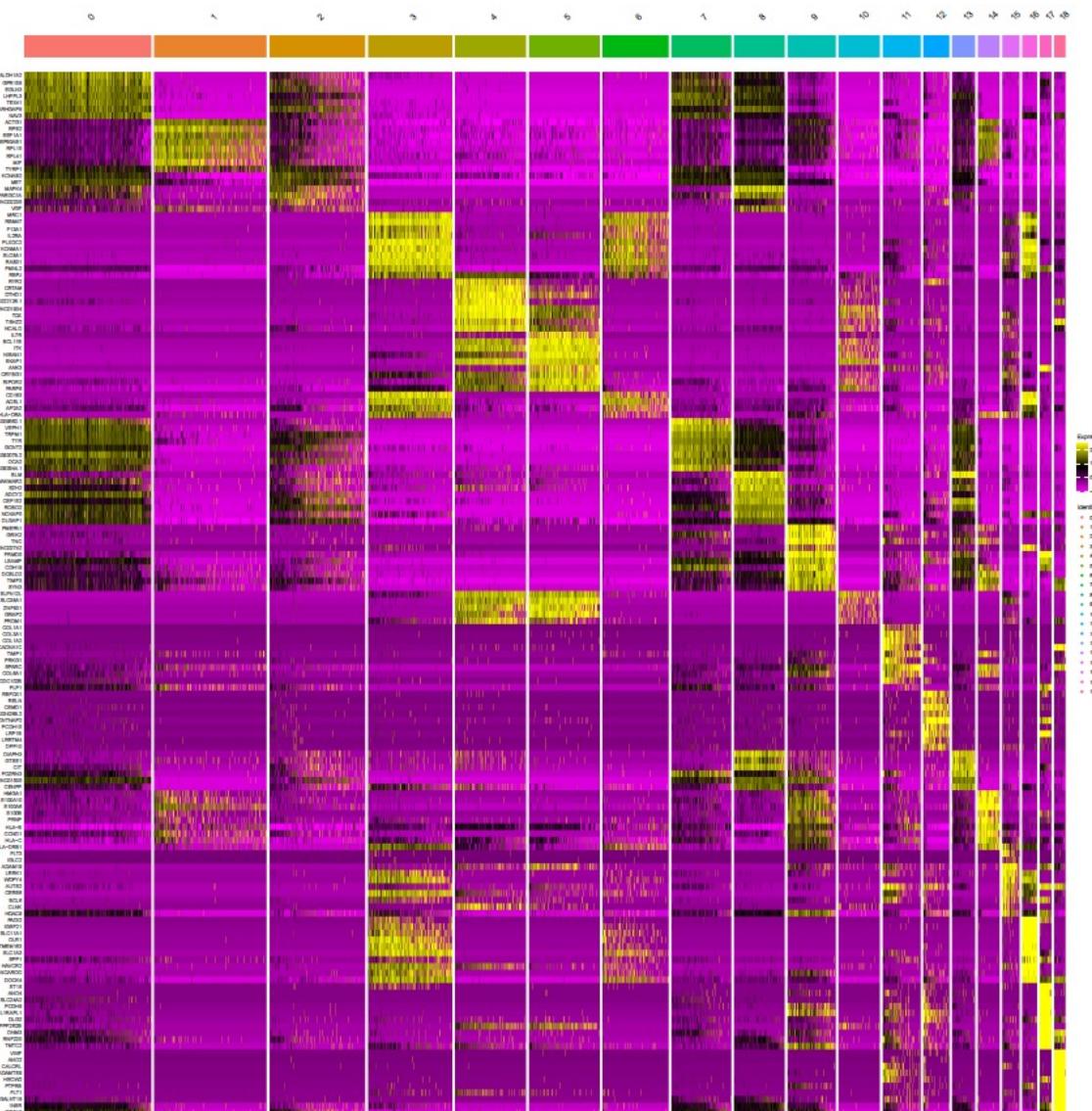
#find markers of differential expression between cluster 1 and cluster 2  
cluster1vs2.markers <- FindMarkers(seu, ident.1 = 1, ident.2 = 2, min.pct =  
0.25)

#find markers of differential expression for all clusters relative to each other  
seu.markers <- FindAllMarkers(seu, only.pos = F, min.pct = 0.25,  
logfc.threshold = 0.25)

#determine top 10 markers in each cluster, ordered by avg_logFC  
top10 = seu.markers %>% group_by(cluster) %>% top_n(n = 10, wt =  
avg_logFC)

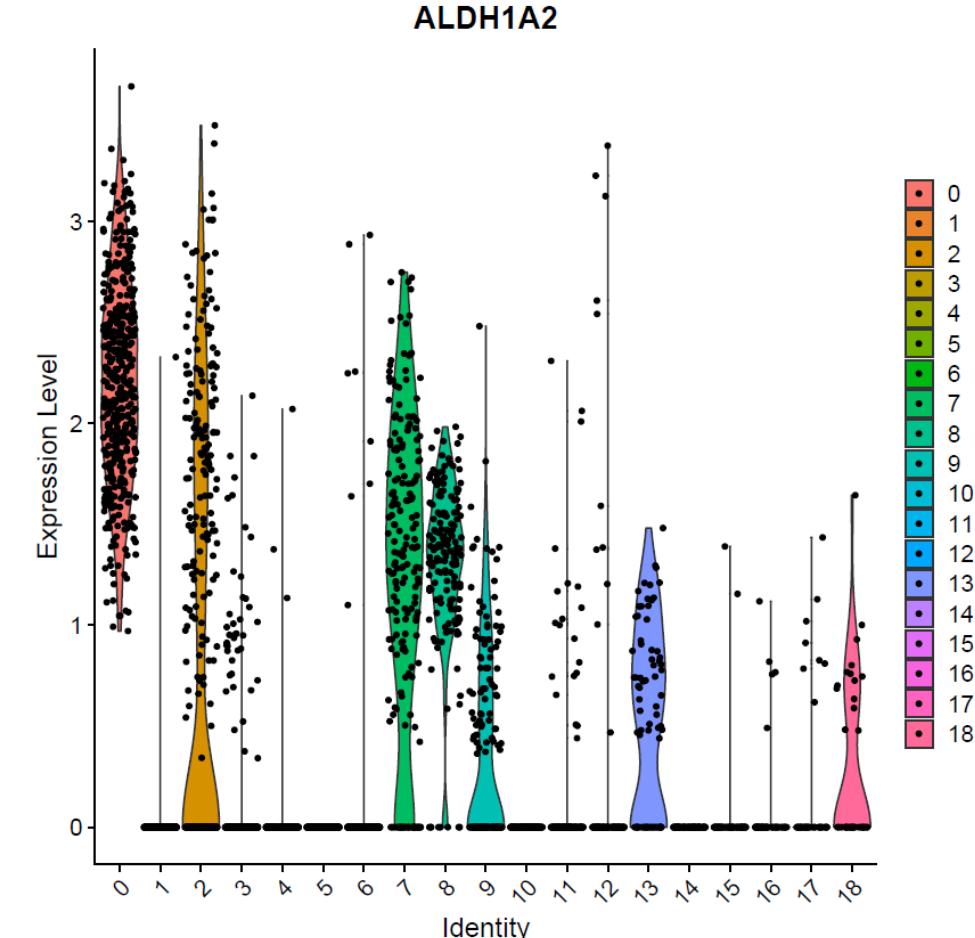
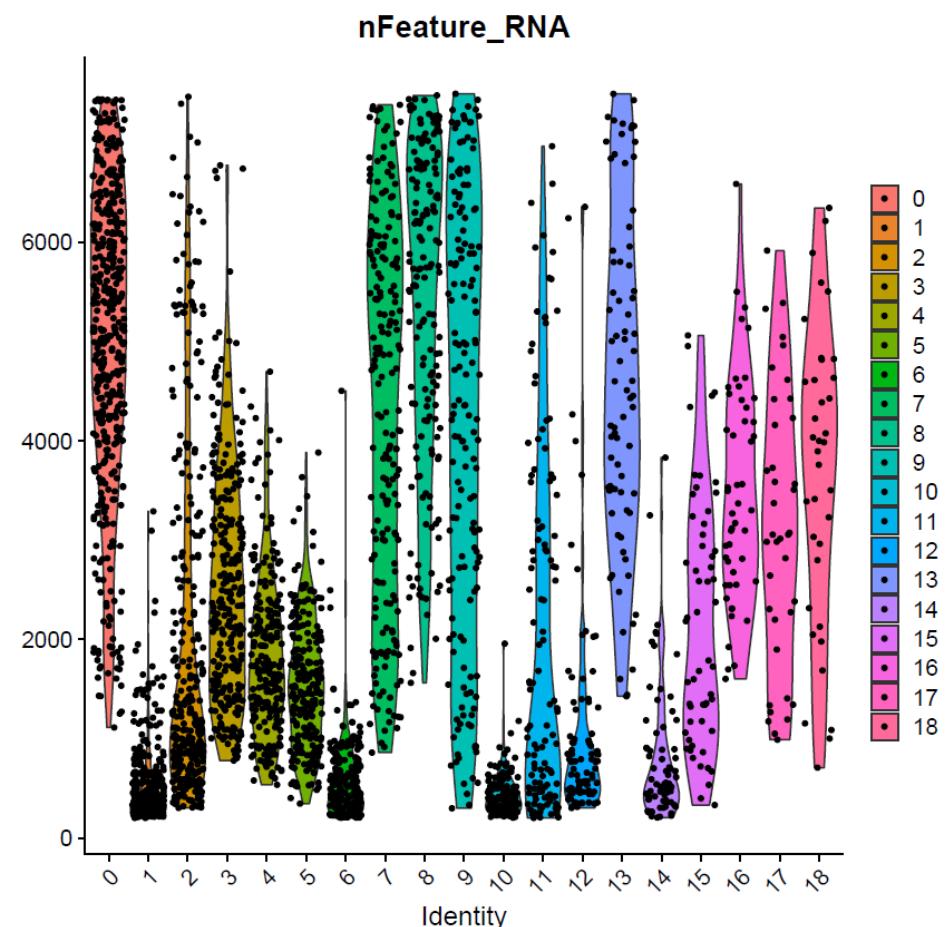
#write top 100 markers to file  
write.csv(top10,  
paste0(directoryname,"/",samplename,"_allmarkers.csv"),row.names = F)

#print heatmap of top 10 cluster gene expression  
#print figure 4x as large (default width and height is 7) to make labels display  
properly  
pdf("workshop_top10_heatmap.pdf",width=28,height=28)  
print(DoHeatmap(seu, features = top10$gene) + NoLegend())  
dev.off()
```



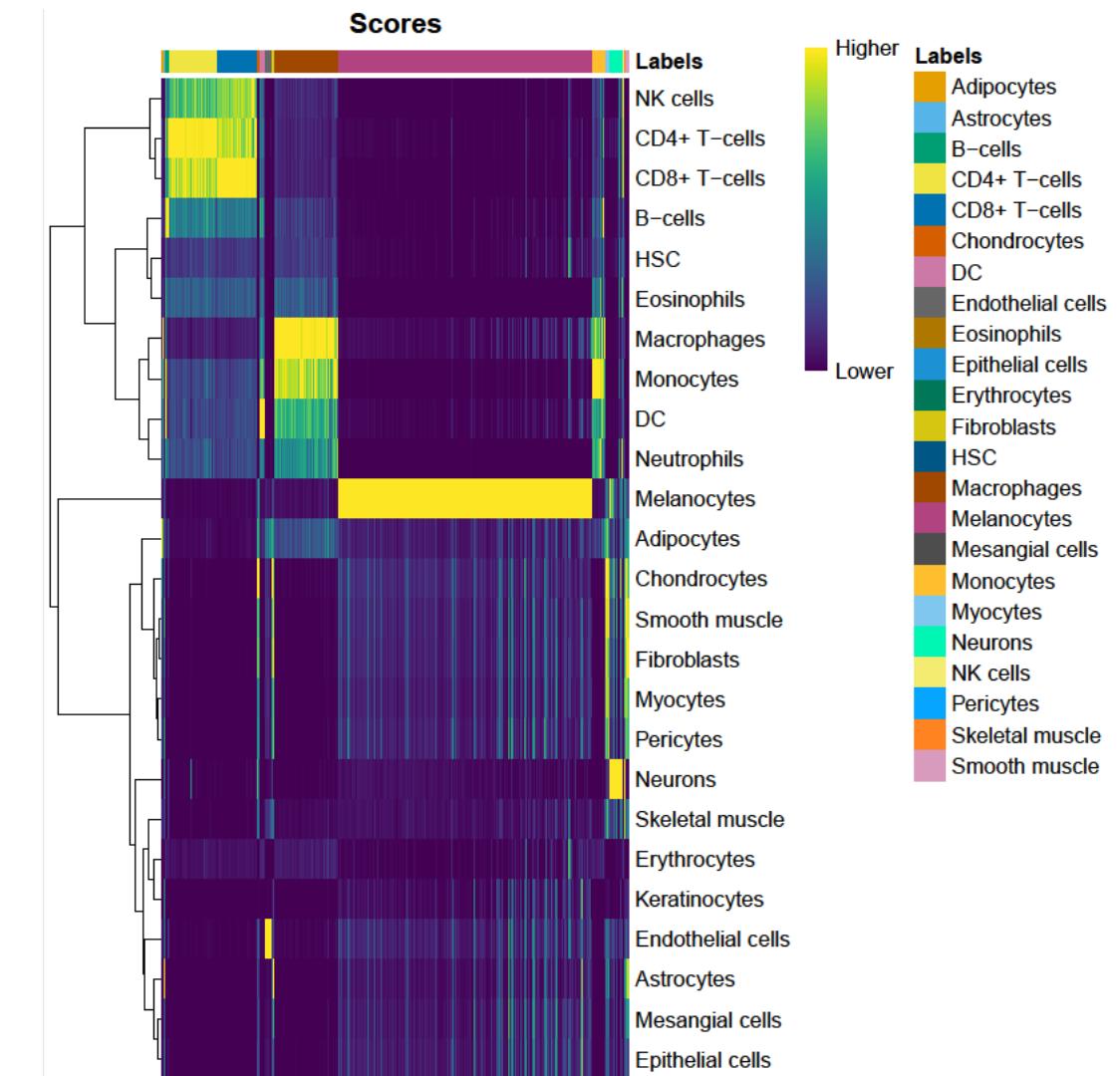
# Print violin plots of quality metrics and gene expression

```
pdf("workshop_vlnplots.pdf")
print(VInPlot(seu, features = c("nFeature_RNA")))
print(VInPlot(seu, features = c("ALDH1A2")))
dev.off()
```



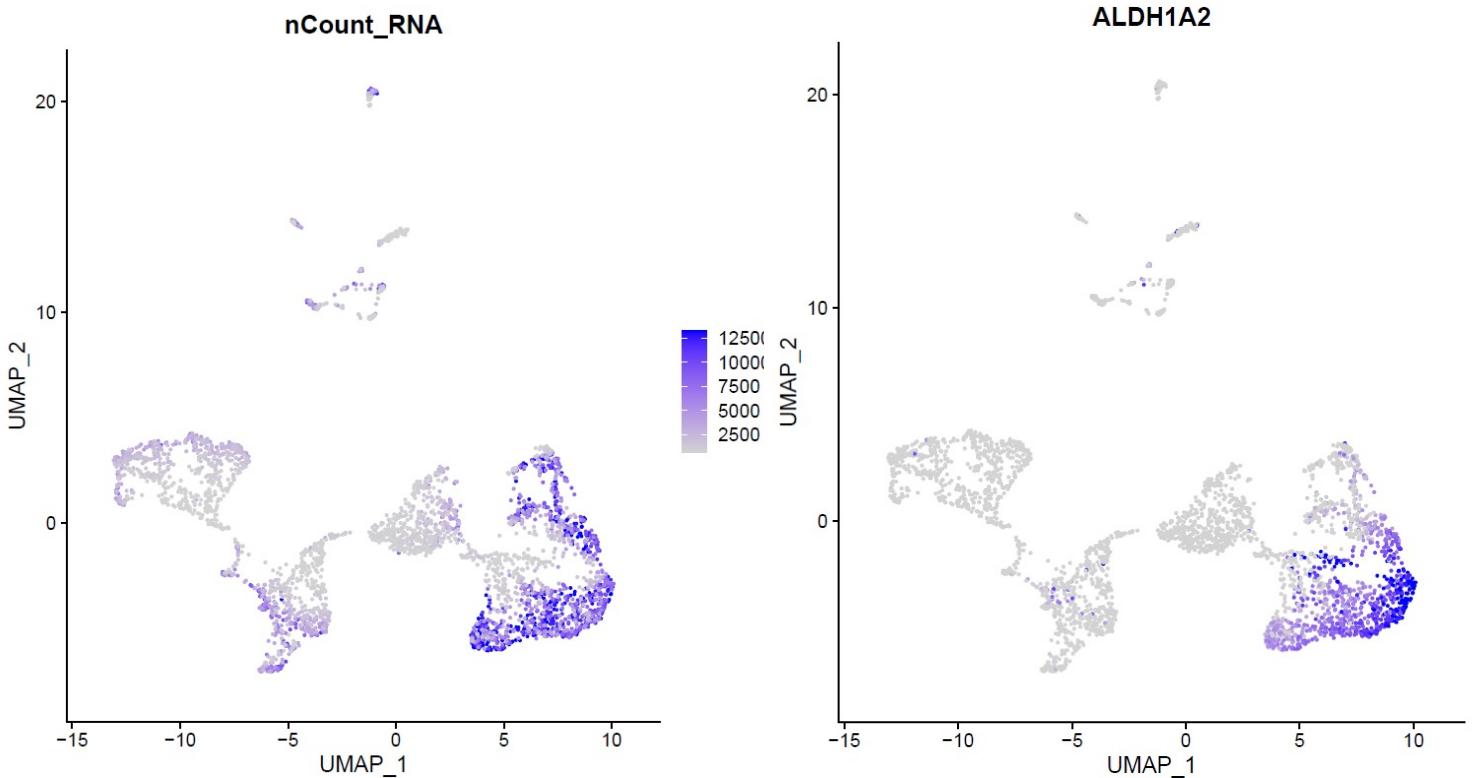
# Annotate single-cell data with cell type classifications, using SingleR reference dataset

```
#load a reference celltype gene expression dataset using BlueprintEncodeData, store it in bped  
bped <- BlueprintEncodeData()  
  
#convert seu to a SingleCellExperiment variable seu_sce, for celltype assignment  
seu_sce <- as.SingleCellExperiment(seu)  
  
#use SingleR to assign predicted celltypes to seu_sce, using bped as a reference  
#label the cells with label.main of bped, which represents  
#the broadest major celltypes in the reference, such as "Epithelial", "Immune", etc.  
pred_bped_main <- SingleR(test = seu_sce, ref = bped, labels = bped$label.main)  
  
#filter predictions in pred_bped_main, then add the predictions to the original seu object as  
celltype_bped_main  
pruneScores(pred_bped_main)  
seu[['celltype_bped_main']] <- pred_bped_main$pruned.labels  
  
#can do the same thing with the label.fine in bped, which represents more fine-grained  
celltypes, and store in celltype_bped_fine  
pred_bped_fine <- SingleR(test = seu_sce, ref = bped, labels = bped$label.fine)  
pruneScores(pred_bped_fine)  
seu[['celltype_bped_fine']] <- pred_bped_fine$pruned.labels  
  
#print heatmap of singleR scores for each cell type  
pdf("workshop_singler_heatmap.pdf")  
plotScoreHeatmap(pred_bped_main)  
dev.off()
```



# Print quality metric and gene expression strength across single-cell data

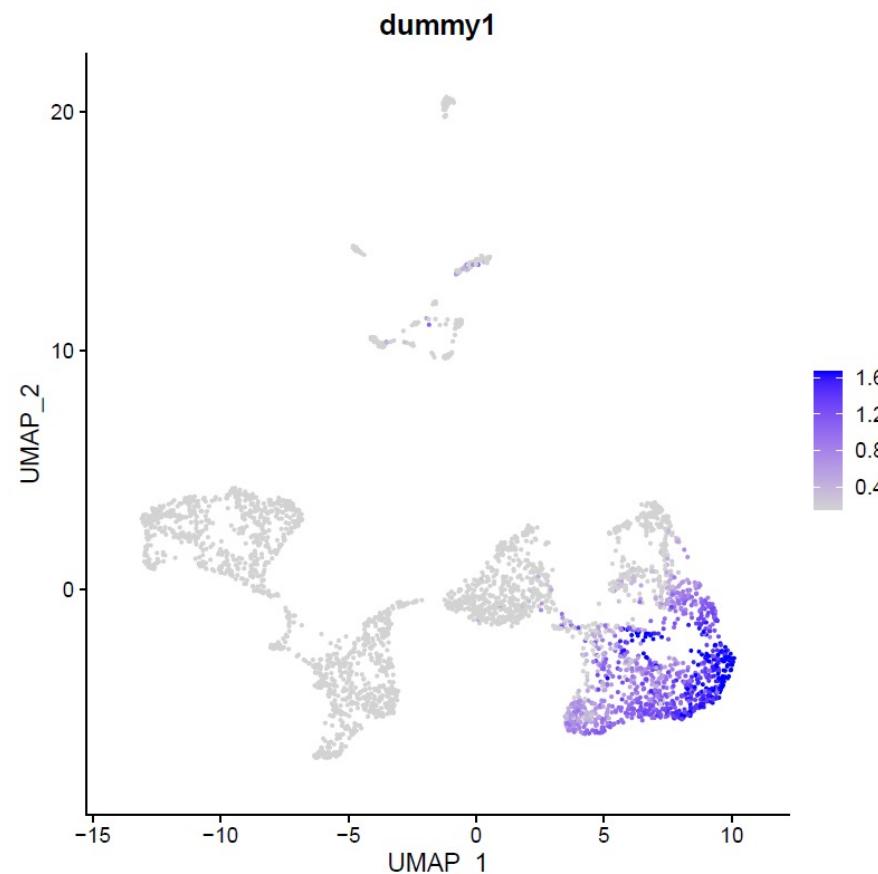
```
#FeaturePlot will plot the strength of each feature in blue across a UMAP  
#stronger features corresponding to more blue  
#features argument takes a list of features to plot  
#features can be a list of quality metrics like percent.mt and nCount_RNA  
#min.cutoff and max.cutoff determine thresholds below and above which not to plot.  
#q9 and q90 indicate 9th and 90th percentiles as cutoffs  
pdf("workshop_featureplot.pdf")  
print(FeaturePlot(seu, features = c("nCount_RNA"),  
min.cutoff = "q9",max.cutoff = "q90"))  
  
#features can also be a list of genes whose expression you want to see on a UMAP  
#some genes may not be in the seu object. To check if a gene is present  
#use "genename" %in% rownames(seu), which returns T if present, and F otherwise  
#then use the following if expression, followed by  
FeaturePlot, to only plot a gene if present in in seu  
if("ALDH1A2" %in% rownames(seu@assays$RNA@data)) {  
print(FeaturePlot(seu, features = c("ALDH1A2"), min.cutoff = "q9",max.cutoff = "q90"))  
}  
dev.off()
```



# Score cells based on strength of a gene expression signature

```
#read dummy_signatures.csv in with read.csv  
#dummy is a set of DE genes in cluster 1 of the data  
#if some signatures are longer than others, pad shorter signatures with  
NA values, using na.strings = ""  
sigs <-  
read.csv(paste0(directoryname,"/dummy_signatures.csv"),na.strings = "")  
#run AddModuleScore for each signature, which will test for the  
signature given in the features argument  
#na.omit(sigs[c]) strips out NA values from each signature  
#the name argument labels the module score in seu with the name of  
the signature  
#however, by default, Seurat will add 1 to the name argument, thus  
name = "Signature" will result in a field called "Signature1" in the seu  
object  
#assay = "RNA", so the module score is calculated on RNA values, and  
search=T, so if gene names in signature do not match exactly, Seurat will  
search for close matches in the data  
for(c in 1:ncol(sigs)){  
  seu<-AddModuleScore(object = seu,features =  
list(na.omit(sigs[,c])),name = colnames(sigs)[c],assay = 'RNA',search=T)  
}
```

```
#FeaturePlot  
pdf("workshop_signature_featureplot.pdf")  
print(FeaturePlot(seu, features = c("dummy1"), min.cutoff =  
"q9",max.cutoff = "q90"))  
dev.off()
```



```
#at any time in session, can save work in file using  
saveRDS(seu, "seuratworkshopsaved.rds")  
#can load work back in with  
seu = loadRDS("seuratworkshopsaved.rds")
```