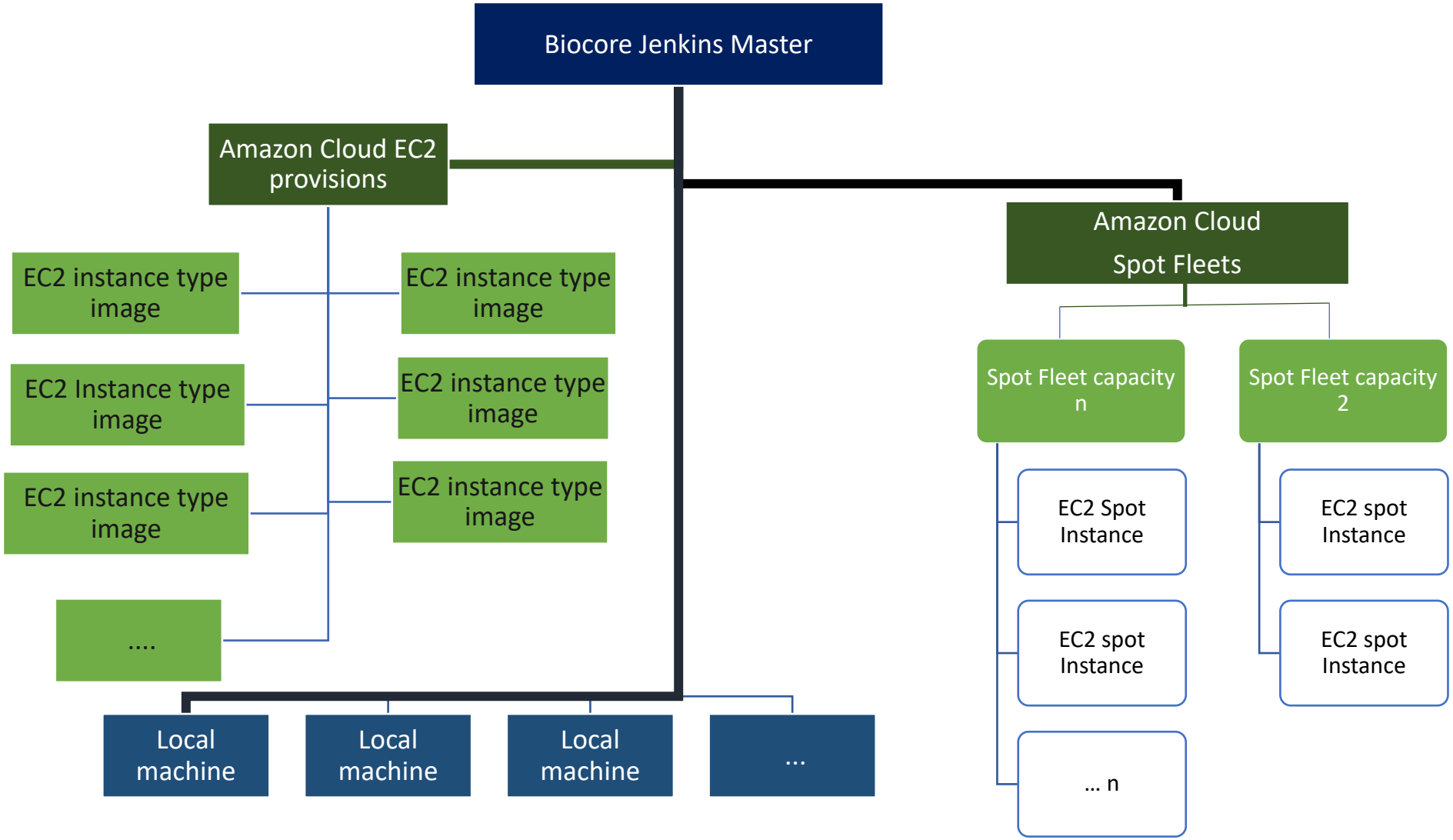
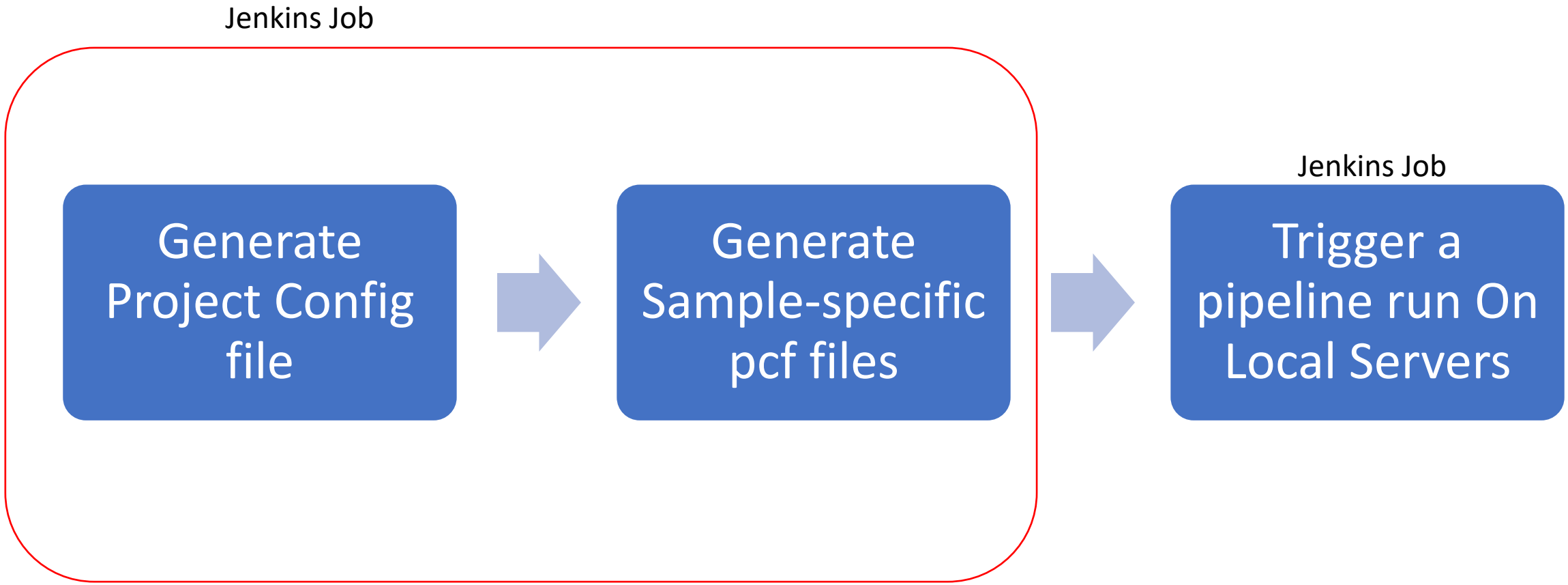
Biocore Pipeline Steps And Standards

Lucie Hutchins

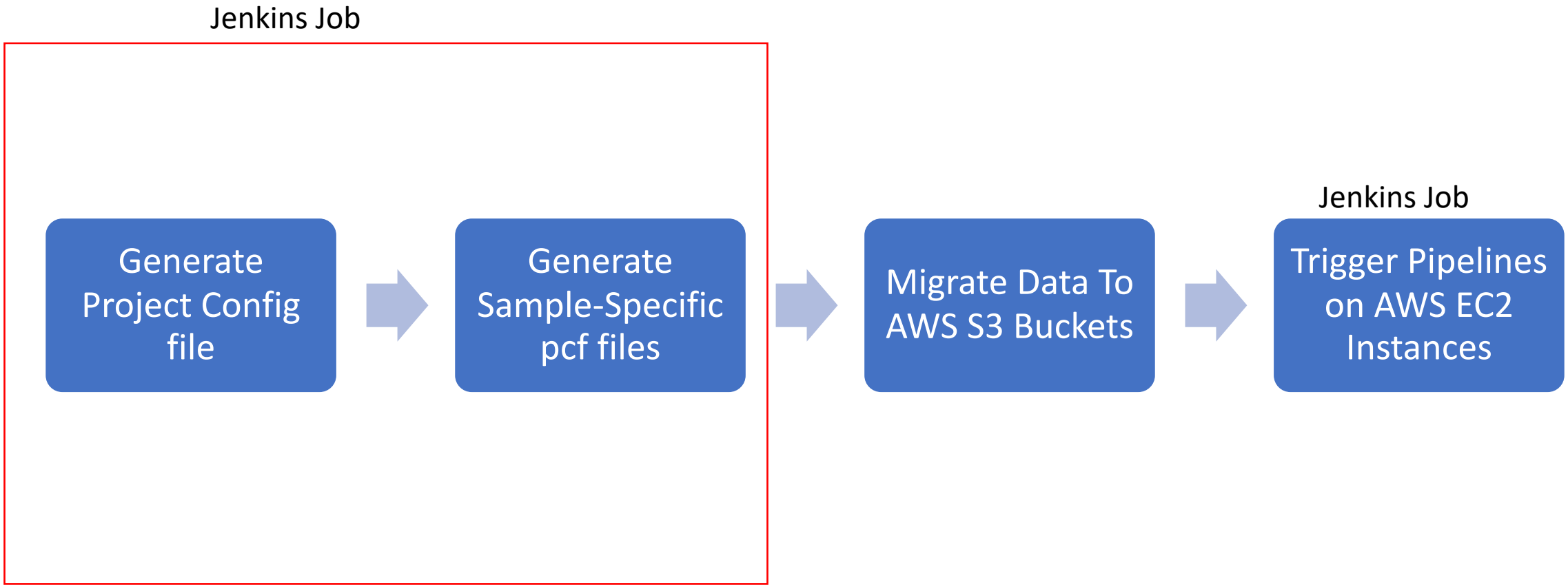
Biocore-Amazon Cloud Hybrid System



Jenkins Workflow – Local Servers



Jenkins Workflow – AWS Cloud Servers



Steps

Scenario: Supposed one of our PIs – say JimCoffman - asked to run a rna-seq project Embryo_Cortisol_2015 through our pipelines, to do this, follow these steps:

1. Download Sequence Reads and create the expected directory structure and naming standards:
   - **Create project directory** – The project's naming format is:
     - **Piusername_linearcounter**.original_project_name (all in lowercase)
     - Example: /data/internal/JimCoffman / **jcoffman _001**.embryo_cortisol_2015
     - You can get PI user names here: https://my.mdibl.org/display/IT/All-Faculty+members
   - **Download sequence reads and Create a design file**
     - **Download sequence reads under** : / data/internal/JimCoffman / **jcoffman _001**.embryo_cortisol_2015
       - Note : Make sure the naming of reads files matches what our system expects - the name of each read file starts with the **sampleID** followed by **readID** …fastq.gz – if not rename the read files (you can use the shell command "mv") – This means the sequence read file name format is a multi-fields name with the first field being the sampleID and the second field the readID(if any –single read samples may oe may not have a readID)
     - **Create a design file for this experiment** :
       - Design file name format: projectName.design.txt under
       - Location: / data/internal/JimCoffman / **jcoffman _001**.embryo_cortisol_2015
       - Note: The design file is a tab-delimited file with the first column of the file storing sample IDs

Steps

Scenario: Supposed one of our PIs – say JimCoffman - asked to run a rna-seq project Embryo_Cortisol_2015 through our pipelines, to do this, follow these steps:

2. Create experiment-wide Cwl file if not exists – Assuming you know how to create one – if not ask for help
3. Un compress the sequence reads under /data/scratch/
   - **Note:** This step is temporary since the current cwl workflow does not work with zipped files
4 Generate experiment/pipeline config files
   - Login to Jenkins and run the "generate-project-config" job
     - **Jenkins** -> **CWL_Workflows** -> **generate-configs** -> **generate-project-config**
       - The above tasks generates the main config file for the entire experiment then triggers the task - **generate-pipeline-pcf** - that generates sample specific pcf files
       - NOTE: **Either one of the above scripts can be ran on the command line as a standalone**
   - **After the above step – check:**
     - **Experiment-level config file** was generated – see Jenkins logs for "generate-project-config" task – expected location : /path2results/JimCoffman/ **jcoffman _001**.embryo_cortisol_2015 / **jcoffman _001** _timestamp/cfgs/pipeline.cfg
     - **Sample-specific pcf files** (sampleID.organism.pcf) are generated – expected location /data/projects/Biocore/biocore_analysis/biocore_projects/pipeline-runs-meta/ /JimCoffman/ **jcoffman _001**.embryo_cortisol_2015 / **jcoffman _001** _timestamp /xxx.danio_rerio.pcf where xxx is the sample ID

Steps

Scenario: Supposed one of our PIs – say JimCoffman asked to run a rna-seq project Embryo_Cortisol_2015 through our pipelines, to do this, follow these steps:

5) Create sample-specific json files –

These files are source controlled - the git repository is biocore_analysis and the path to json files within the repository is biocore_analysis/biocore_projects/rna-seq/ JimCoffman / **jcoffman _001**.embryo_cortisol_2015 /**jcoffman _001** _timestamp We install this repository under /data/projects/Biocore/

- Create Json template under /path2results/teamName/projectName/runID/cfgs/template.json
- Create sample-specific json file using this template

Steps

Scenario: Supposed one of our PIs – say JimCoffman asked to run a rna-seq project Embryo_Cortisol_2015 through our pipelines, to do this, follow these steps:

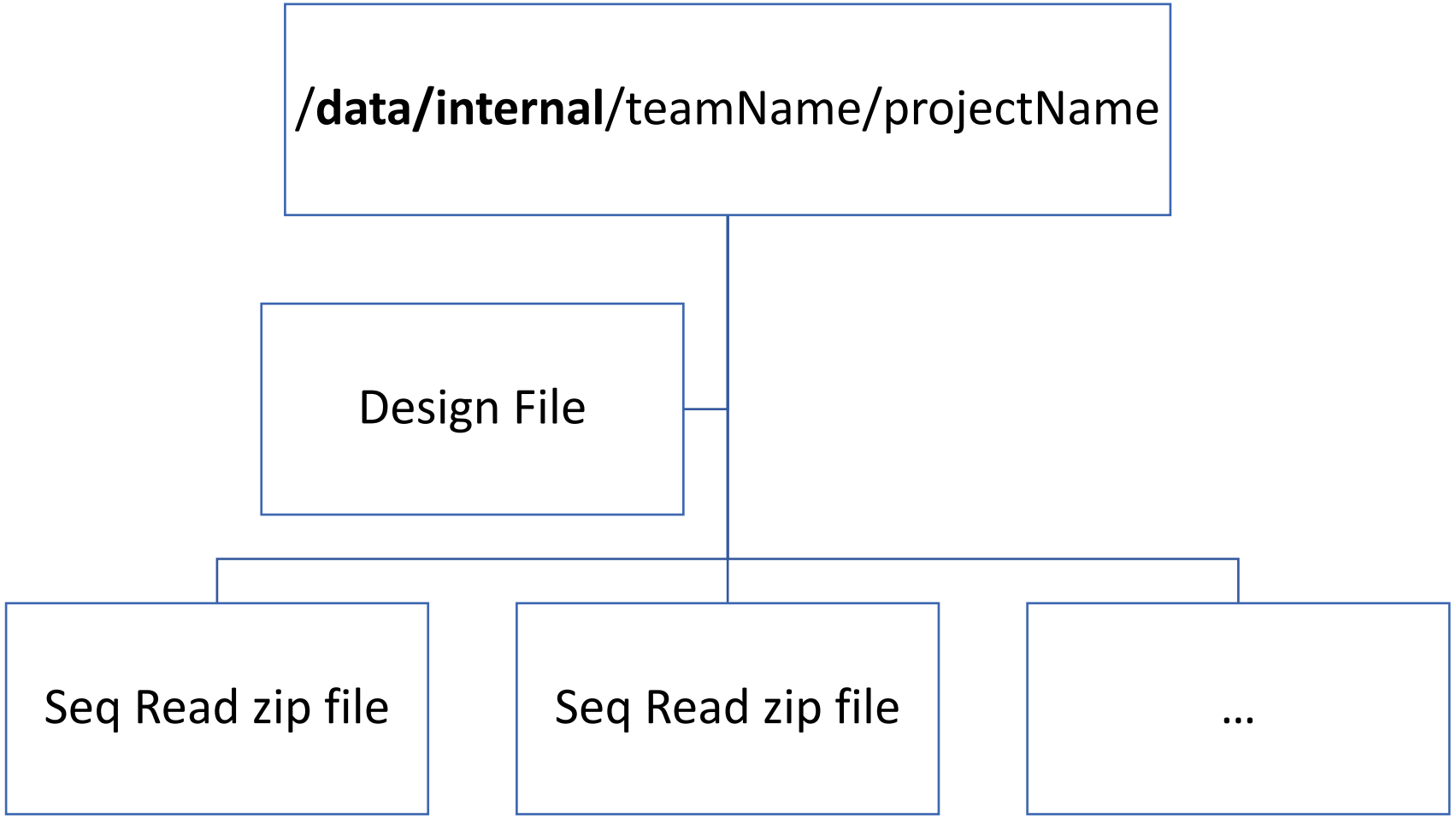
6 Migrate this project's data to the cloud: (only when running pipelines on cloud)

- Migrate the cwl script to corresponding efs mapping (Currently the ggr-cwl package --)
- Migrate json files to corresponding location on S3 bucket
- Migrate pcf files to corresponding location on S3 bucket
- Migrate sequence reads to corresponding location on S3 bucket
- Migrate the Ref /data/scratch/ensemble-93/danio_rerio* to corresponding S3 bucket
- Migrate RSEM and STAR indexes :
  - /data/transformed/RSEM…/ensemble-93/danio_rerio to the corresponding S3 bucket
  - /data/transformed/STAR…/ensemble-93/danio_rerio to the corresponding S3 bucket

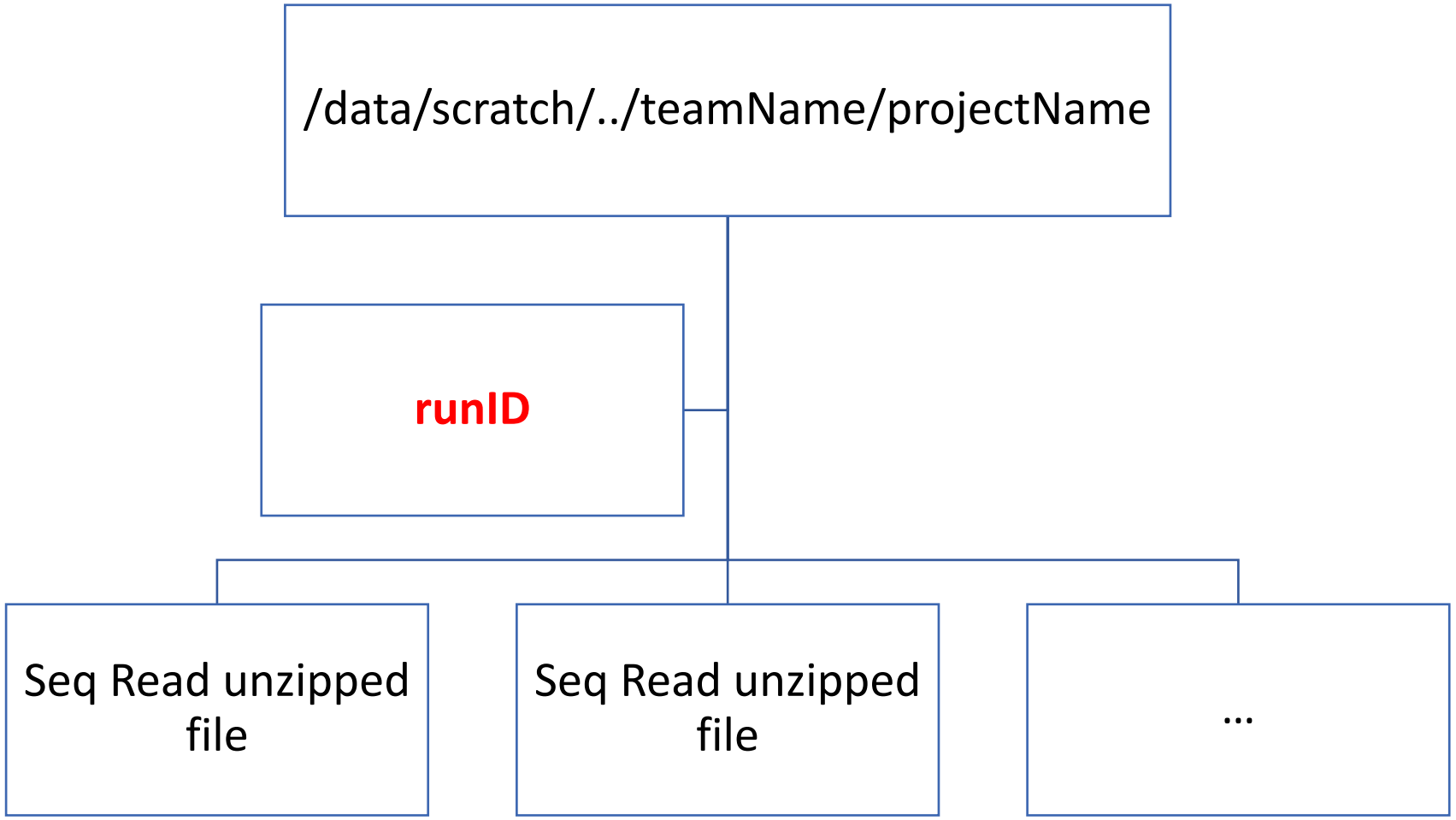
7 Trigger the pipeline Run – On Jenkins

- On Local servers
  - **Single–sample pipeline**: run cwl_workflows/**single-pipeline-local**/run-pipeline
  - **Multi-sample pipelines**: run cwl_workflows/multiple-pipelines-in-parallel with server type set to "local"
- On Cloud servers
  - **Single-sample pipeline** : run cwl_workflows/single-pipeline-cloud/run-pipeline-cloud
  - **Multi-sample pipelines**: run cwl_workflows/multiple-pipelines-in-parallel with server type set to "cloud"

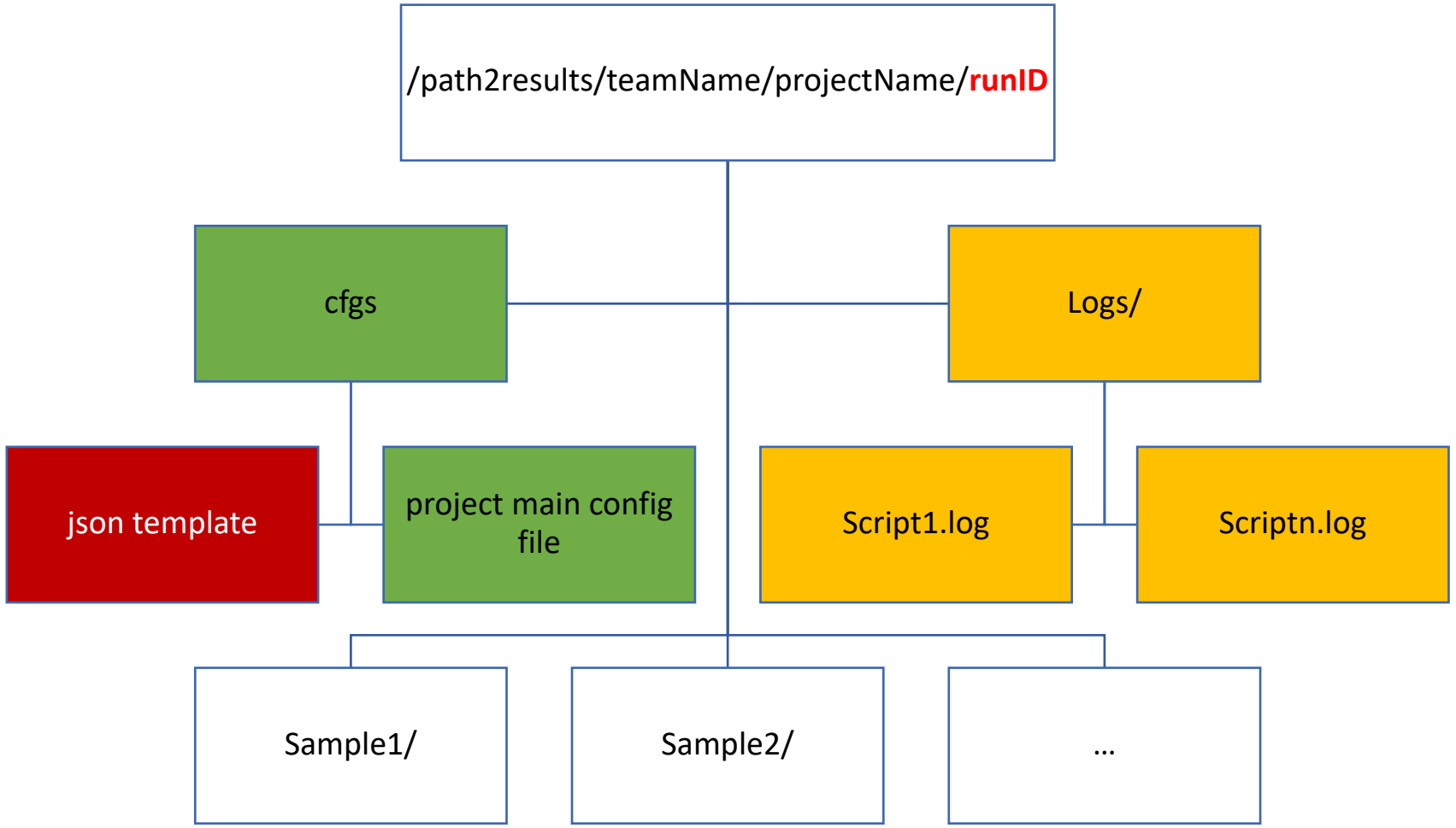
Expected: Original Reads



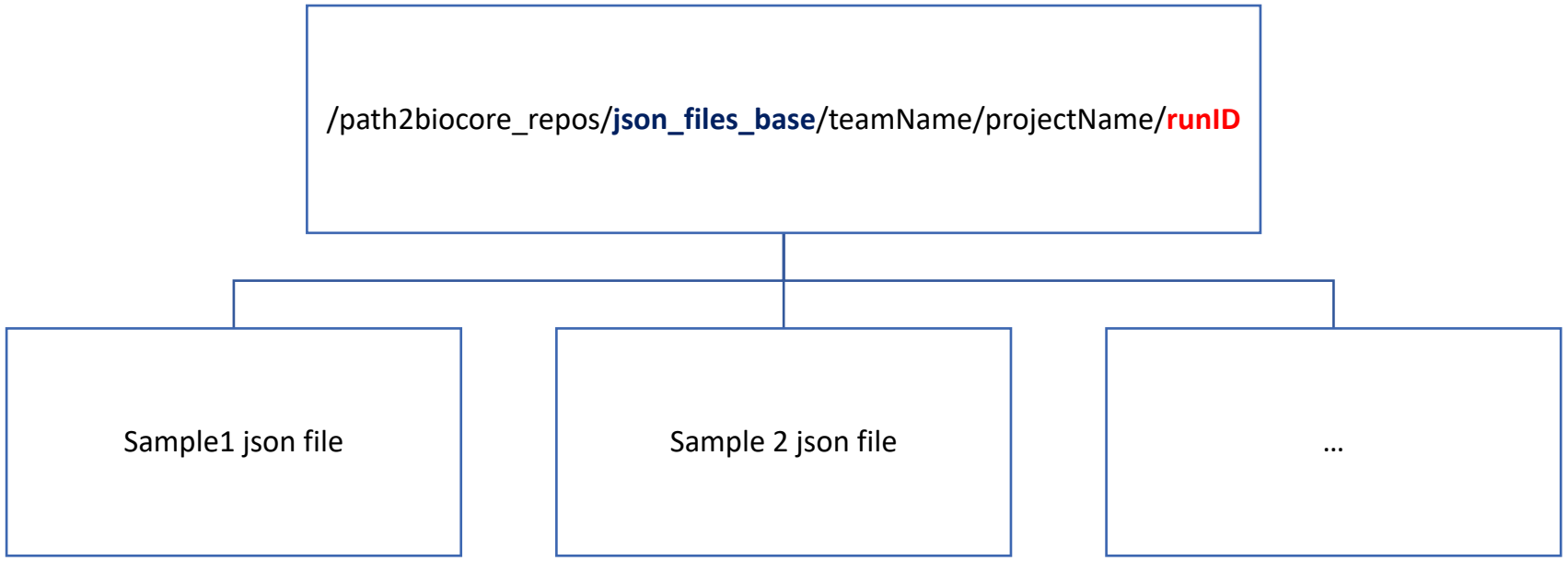
Expected: Scratch unzipped reads



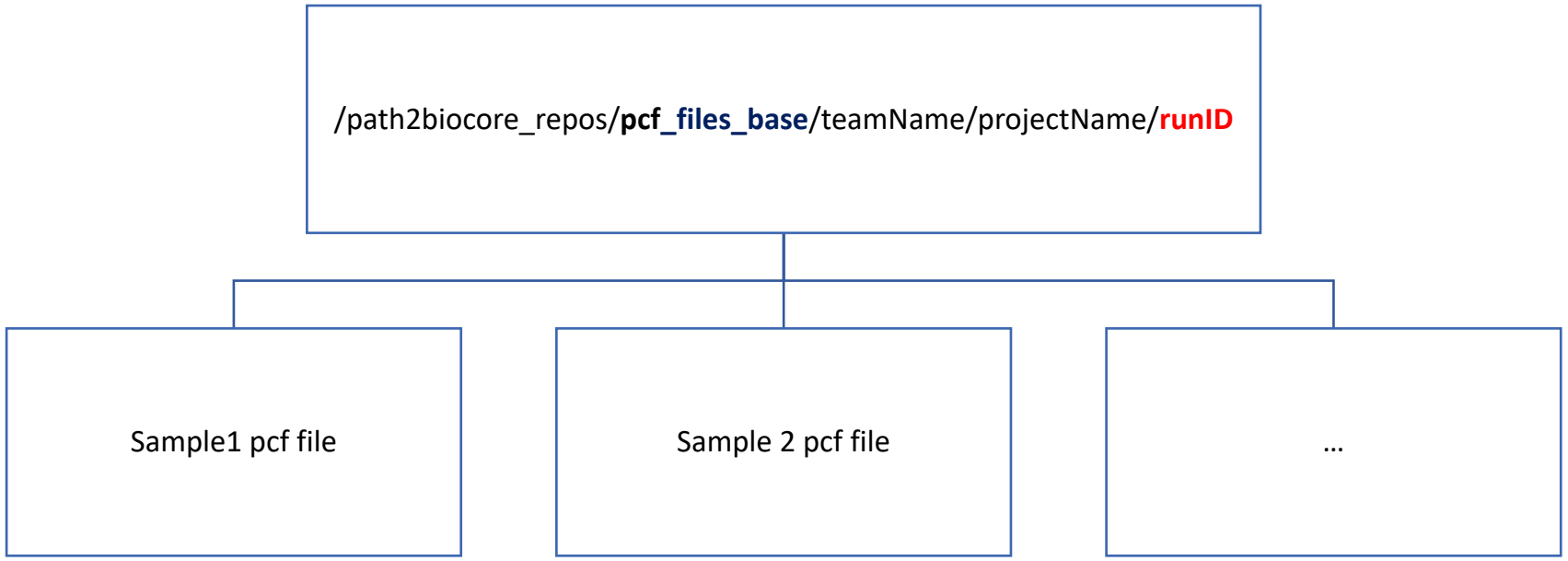
Expected: Results Dir Structure



Expected: Json files



Expected: PCF Files



Expected:

- **Project:**
  - **Name Format:**
    - **Piusername_linearcounter**.original_project_name (all in lowercase)
    - Example: /data/internal/JimCoffman / **jcoffman _001**.embryo_cortisol_2015
    - You can get PI user names here: https://my.mdibl.org/display/IT/All-Faculty+members
- **Reads**
  - **Location** /data/internal/TeamName/ProjectName/
  - **Read file name format**
    - sampleID[delimiter]readID[delimiter] ..fastq.*
- **Design File**:
  - Name format:
    - ProjectName.design.txt
  - Location: /data/internal/TeamName/ProjectName/

Expected:

- **Intermediary results – if needed**
  - **Location:** /data/scratch/…/TeamName/ProjectName/ProjectPrefix_timestamp/
  - **Structure under the results root directory:**
    - cfgs/pipeline.cfg
    - logs/
    - Sample1/
    - …..
    - Sample..n/
    - Results are stored by runs (projectPrefix_timestamp) – and by sample
- **Final results**
  - /data/projects/TeamName/ProjectName/ProjectPrefix_timestamp/

Expected

- **Pipeline Json files:**
  - **Location:**
    - /data/projects/Biocore/**biocore_analysis**/biocore_projects/…/TeamName/ProjectName/ ProjectPrefix_timestamp/
  - **File name format:**
    - sampleID1.organism.json
    - ….
    - sampleIDn.organism.json
- **Pipeline pcf files :**
  - **Location:**
  - /data/projects/Biocore/**biocore_analysis**/biocore_projects/pipeline-runs-meta/TeamName/ProjectName/ ProjectPrefix_timestamp/
  - **File name format:**
    - sampleID1.organism.pcf
    - ….
    - sampleIDn.organism.pcf

Biocore Jenkins –AWS Cloud Hybrid system

- I configured our local Jenkins server with the following:
  - To launch Amazon Machine images (AMIs) as Slaves: 7
    - General purpose EC2 instances –
      - M4.4xlarge: 16vcpus, 64GB ram, 600GB EBS
      - M4.4xlarge: 16vcpus, 64GB ram, 800GB EBS
      - M5.4xlarge: 16vcpus, 64GB ram, 800GB EBS
    - Memory Optimized EC2 instances
      - R4.4xlarge: 16vcpus, 122GB ram, 600GB EBS
      - R5.4xlarge: 16vcpus, 128GB ram, 600GB EBS
      - R4.4xlarge: 16vcpus, 122GB ram, 800GB EBS
      - R5.4xlarge: 16vcpus, 128GB ram, 800GB EBS
    - We have the flexibility to add machine images as needed
  - To launch Spot Fleets as slaves –
    - Jenkins automatically schedules all EC2 instances from any active fleet
    - Jenkins would scale the fleet capacity down or up depending on the workload

Jenkins –Cloud Setting

Amazon Machine images (AMIs):

- Spot instances are requested by default – however, Jenkins falls back to on-demand instances if the spot request was not filled.
- The idle termination time for any EC2 instance launched by Jenkins is set to 30 minutes – meaning, if an active machine is idle for 30 minutes then Jenkins will shut the instance down.
- All EC2 instances are EBS optimized
- All EC2 instances are Linux servers
- Machines are only launched when a new workload is detected

Spot Fleet:

- Jenkins automatically detects and schedules all running instances of an active Spot Fleet – In addition, Jenkins would scale up and down the number of slaves depending on the demand – However, Jenkins does not terminate these instances.

Jenkins –Cloud Setting

Spot Instances:

- Couple weeks ago, in one of Amazon Cloud's free seminars on Spot instances (auction-based cloud virtual machines), the presenter said there is a 95% chance that your spot instance will complete the job before being terminated and assigned to an on-demand request - While this may be true in many cases, I find it hard to believe since my experience shows otherwise. Out of 12 spot instances launched this week to run the rna-seq pipelines, only one ran to completion. Some instances were terminated after few minutes running and some after few hours. Spot Instances are available at up to a 90% discount compared to On-Demand prices. However, Spot instances have the lowest priority compared to On-Demand instances. This means that if a new request for On-Demand instance and there is none in the pool of instances, your spot instances is more likely to be killed - to satisfy the On-Demand request - even if your job is still running.

NOTES

- Project's Naming - Joel's Specs:
  - **Piusername**_xxx.project_original_name - all in lowercase
  - Where:
    - Piusername is the MDIBL user name for this PI (
      - where do we get this?
        - See: https://my.mdibl.org/display/IT/All-Faculty+members
      - What if the PI does not have a user name?
        - Create one
    - XXX is what?
      - A linear count – example: 001 , 002, 003 , …
- Project's Design Files - Joel's Specs:
  - This needs quite a bit more fleshing out; the design file needs much more than that, and has some constraints on it that are put in place by other programs that use it (DESeq2, for example)

NOTES

- Sequence Read File's Naming
  - Joel's Specs:
    - The renaming needs to be done in a controlled manner- manual renaming such as we did for the Coffman data is a last resort that is far from ideal. Ideally, we will write some monitoring/data qc scripts that will identify "non-standard" or unacceptable naming, and then reformat into an accepted version.
  - Current format: a multi-field name:
    - SampleID[delimiter]ReadID[delimiter][….].gz
- Pipeline Results: Joel's Specs:
  - I need to be convinced still why we are storing results under /data/scratch/rna-seq/Team/Project/results/.
  - Why are we not storing directly in /data/projects/Team/Project/?
  - That is the final ending place for the results, so why create the need for a separate copying action after completion?
  - I do understand that /data/scratch/rna-seq was set up as a place for the decompressed fastq files, but there's no reason of which I'm aware that the results need to go there before going into /data/projects/.
  - If I am missing something, please explain, but otherwise, please see if you can make the results go straight to the /data/projects/ hierarchy
- Pipeline Results: Lucie's Notes:
  - As currently designed, the base location of the output is specified at run time as input - so this is not hardcoded. The program uses the base location in addition to the teamName and projectName to create the expected structure.