Nextflow on AWS Batch User Guide

Introduction

This is an introductory guide for the genomics secondary analysis solution based on the Nextflow on AWS Batch framework. This is an AWS-developed solution that runs Nextflow workflows in a cloud-based environment, orchestrated by AWS Batch. This guide is intended for end users who are starting to use an existing deployment; please read the full documentation for Nextflow on AWS Batch for a complete guide.

This guide assumes the following:

- Familiarity with Nextflow, including the creation and editing of configuration files.
- Experience with working at the command line, such as with bash.
- Access to an AWS account where the Nextflow solution has been deployed. The account will be accessed through the AWS Console, and also from the command line.

Workflow

This solution uses AWS Batch to provide the compute resources for a Nextflow pipeline running containerized Docker images. A Nextflow run is controlled by a single Nextflow **Head Job**, which runs a number of **Task Jobs** to complete the processing pipeline on each input file. The head job runs a containerized Nextflow executable, and remains running for the duration of the entire run. Each task job performs one step of processing (such as STAR alignment) on one input file, and runs only for the duration of that step. A Nextflow processing pipeline of 4 steps, run on 10 input files, will run one long-duration head job and 40 shorter-term task jobs. The Docker image for the head job is provided by the solution, and the images for the task jobs are defined in the Nextflow configuration, and can be pulled from public sources such as quay.io or Dockerhub.

The configuration files for the Nextflow head job are stored in an Amazon S3 bucket, and are provided as a parameter to the overall Nextflow run. The Nextflow task jobs also access S3 buckets for input (eg fastq) and output (eg bam) files. The names of the buckets, and the paths within them, are defined in the Nextflow configuration file. It is not necessary to use separate buckets for configuration, input, and output: you can use a single bucket containing paths (folders) such as `config`, `data`, and `results`.

AWS Batch

AWS Batch is a web service that runs multiple container-based compute jobs as Docker processes running on Amazon EC2 compute instances. Batch handles the deployment and scaling of the compute resources required for the jobs in the Nextflow pipeline, and ensures that all jobs are run to completion.

A Batch **job** is a single task to be performed. The Nextflow head job, and all task jobs, are run as individual jobs in Batch. Each job is self-contained and performs one job, by downloading and running a Docker container image. Upon completion, the task saves its output and terminates, signalling Batch to start the next step in the pipeline, based upon the output files from the previous step.

Batch jobs are submitted to **queues**, from where they are allocated to compute resources as they become available upon completion of a previous task. This solution uses two queues in which to run jobs: a **High Priority** queue and a **Default** queue. The Nextflow head job runs in the high priority queue, which ensures that it is not terminated during the Nextflow run. The Nextflow task jobs are submitted by the head job into the default queue, which allows termination and re-submission of the jobs in the queue.

Compute Resources

Amazon EC2 provides the compute resources (virtual machine instances) on which Docker is run to perform the task processing. This process is transparent: Batch starts and initializes the EC2 instances, which makes them available to receive and process a job from the queue. When there are no more jobs queued for processing, Batch shuts down the instances.

EC2 instances are provisioned as either **On-Demand** or **Spot** instances. On-demand instances are dedicated to your use, and run until Batch explicitly shuts them down. Jobs in the high priority queue (the Nextflow head job) run on on-demand instances, as this job must remain running for the duration of the run.

Spot instances cost less than on-demand instances, as they make use of excess capacity in the EC2 service, but they may be interrupted and recalled by AWS. The majority of the overall processing is done by task jobs, submitted by the Nextflow head job, and these are run in the default queue on Spot instances. Batch handles the resubmission of any jobs in the default queue that are interrupted before they complete.

Summary of Steps

The steps involved in running a Nextflow job on AWS are as follows. Further explanation of the components involved follows below.

- Deploy a Nextflow infrastructure stack: In this guide we assume this has been performed. Required from the deployment's output are the identifiers of the Job Queue and Job Definition created by Batch, which are needed in the Batch configuration file.
- Install and test the AWS Command Line Interface (CLI) tools, to make the `aws` command available from your command line. From the command line you will submit the Nextflow job to the Batch environment.
- If not using the bucket created by the solution (such as to use existing buckets containing data) create S3 buckets for the Nextflow configuration, input, and output files, and create any paths necessary within the buckets.
- Upload your Nextflow configuration files (ie `main.nf`, `nextflow.config`) to the Nextflow configuration bucket.
- Upload the fastq input files to the input bucket.
- Create a Batch configuration file containing the job queue and definition names, and the location of the Nextflow configuration files.
- Submit your Nextflow run to AWS Batch by either running `aws batch` on the command line and specifying the Batch configuration file, or by writing a script to perform this step.
- After processing, the results files are stored in S3 for further processing, and the Nextflow reports may be downloaded for analysis.

Batch and Nextflow Configuration

The configuration file required to submit the Nextflow run contains parameters that are specific to Batch, and specific to Nextflow.

The Batch-specific parameters are two Batch resources created by the infrastructure stack:

- **Job Queue**: The identifier of the queue to which to submit the Nextflow head job
- **Job Definition**: The name of the configuration of the Nextflow head job

The Nextflow-specific parameters defined in the **containerOverrides** section of the configuration file:

- Location of your Nextflow configuration in S3, ie `s3://mybucket/myworkflow/`
- **reads** parameter: S3 bucket and path, and names of your input (ie fastq) files
- **outDir** parameter: Location of the output S3 bucket and path
- **with-report**, **with-timeline** parameters: S3 bucket, path, and name of output reports

Note that the **reads** parameter is in a unique S3-like syntax that also contains wildcards: `s3://mybucket/mydatapath/mydata*_{1,2}.fastq.gz`. This differs from standard S3 syntax where only prefixes are defined: `s3://mybucket/mydatapath/mydata`

A configuration file in JSON format, to submit to AWS Batch may look like the following:

```
{
    "jobName": "nextflow-job-00",
     "jobQueue": "arn:aws:batch:us-east-1:0123:job-queue/highpriority-f00",
     "jobDefinition": "nextflow-my-job-definition",
     "containerOverrides": {
        "command": [
          "s3://my-bucket/nextflow-config/",
          "--reads", "s3://my-bucket/input-data/00*_{1,2}.fastq.gz",
          "-profile", "awsbatch",
          "--outDir",      "s3://my-bucket/results/",
          "-with-report",  "s3://my-bucket/results/my-report.html",
          "-with-timeline","s3://my-bucket/results/my-timeline.html",
          "--alignmentmethod","STAR"
        ]
    }
}
```

Testing the Infrastructure Stack

A full run of Nextflow involving multiple steps performed on each of multiple input files, is a complex and lengthy process. Smaller tests can be run to test individual components of the Nextflow infrastructure stack.

AWS CLI Connectivity

The first step is to ensure that you are able to access the correct AWS account from the command line. After installing and configuring the AWS CLI, issue the command `aws sts get-caller-identity` to check that you have access to the correct account (the account number will be returned in the output). If you have named profiles configured for your AWS CLI (for access to multiple accounts), you may need to specify `--profile MYPROFILE` in the command. If you use a default profile, or only one profile, this can be omitted.

Hello World

This test provides a test of connectivity to AWS Batch. It will validate that you have the **aws** command line tools installed, have access to the correct account, can start a Batch job, and have the correct job queue and job definition. This job starts one head job in the high priority queue, which in turns starts four jobs in the default queue, each of which prints

'Hello, World'. This job runs the Nextflow container image but does not perform any genomic processing.

```
aws --profile PROFILE batch submit-job --job-name JOB-NAME --job-queue HIGH-
PRIORITY-QUEUE --job-definition JOB-DEFINITION --container-overrides
command=hello
```

Submitting this command should produce output similar to:

```
{
    "jobArn": "arn:aws:batch:us-east-1:012345678901:job/a791dcef-f00-f00",
    "jobName": "nextflow-hello-00",
    "jobId": "a791dcef-f00-f00"
}
```

The AWS Batch console will now show the 'hello' main job in the high priority queue, and this job will in turn run four 'sayHello' jobs in the default queue. The progression of the jobs through the queues is described in 'Monitoring', below.

Nextflow Minimal Test

This test is a script that performs a small genomics processing task by launching a Nextflow workflow downloaded from GitHub. This is a more comprehensive test than the 'hello world' test, performing a full processing pipeline on public genomic data.

Download the script `nextflow-minimal-test.sh` from https://github.com/crabba/nextflow-test and run it on the command line:

- `nextflow-minimal-test.sh` will display help text
- `nextflow-minimal-test.sh <profile>` will display the available high priority queue and job definition for the given AWS CLI profile.
- `nextflow-minimal-test.sh <profile> <queue_name> <job_definition>` will submit the job.

This script will download and run wleepang/demo-genomics-workflow-nextflow, which in turn will download and run several biocontainers container images. Input and reference sequence files are downloaded from public S3 buckets. The progress of the test can be monitored as described below, and its success shown by all jobs completing in the

'Succeeded' state.

Submitting a Nextflow Job

The exact method of submitting a Nextflow job will depend upon your configuration. In this example, once you have uploaded your Nextflow configuration files to an S3 bucket and path (`my-bucket/nextflow-config/` in the example above), you can submit the job to Batch using the sample JSON-format configuration file:

```
aws --profile MYPROFILE batch submit-job --cli-input-json file://my-config-
file.json
```

Once submitted, the progress of the Nextflow head job and task jobs can be monitored from the Batch dashboard.

Monitoring

The AWS Batch console will now show the main (Nextflow) job in the high priority queue, and it will move through the stages of the queue (Submitted ➡ Pending ➡ Runnable ➡ Starting) until it is in Running state. This process takes several minutes, while the EC2 instance hosting the job is started. Once the main job is running, it will submit task jobs to the default priority queue, each of which will also progress through the same sequence, stop upon completion, and should end in 'Succeeded' state. Once all task jobs have completed, the head job will stop and move to 'Succeeded' state.

While the job is running, you can examine the underlying instances in the EC2 console, as they are launched, initialized, run, and then shut down. The head job will run on an on-demand instance and the task jobs will run on Spot instances. The initial deployment of EC2 instances will take a few minutes, though once they are running, they can immediately accept a new job after completion of the previous job, without shutting down in between.

A range of EC2 instance types (eg m5.4xlarge, r5.2xlarge) is configured into the Batch environment, to provide the maximum availability of Spot instances. Each EC2 instance can host multiple task jobs, the number of which depends upon the size of the instance and the CPU and memory requirements of the job. The maximum number of instances running at once is defined by the total CPU limit, which is configured into Batch.

Once each job has completed, you can examine its log files. The `/aws/batch/job` log group in CloudWatch will hold the logs of the head job as well as for each of the task jobs.

Troubleshooting

If you do not get a jobId returned when you submit a Nextflow job, check that you have CLI connectivity and that you have used the correct high–priority queue identifier and job description. You can retrieve the identifiers of the high priority queue and job definition from the AWS Batch console, or by issuing the CLI commands `aws batch describe-job-queues` and `aws batch describe-job-definitions --status ACTIVE`

Once the job has been submitted, within a few minutes you should see the job in the high priority queue transition to 'Runnable' state, and in the EC2 console you will see an on-demand instance starting up to serve the Nextflow head job. From this point, the head job will start logging its activity to CloudWatch, and you can access the log of each job by clicking on 'Log Stream Name' in the details of the job in the Batch console.

The Nextflow tasks will log results to CloudWatch. These logs are useful for debugging and for monitoring the flow of data. While the naming of the CloudWatch log groups depends upon configuration, the log groups typically contain the name of the Nextflow stack, which also forms part of the name of the high priority queue.

After the on-demand instance is running, its Nextflow task will submit to the default Batch queue the task jobs which will follow a similar progression and start up Spot EC2 instances. After the Spot EC2 instances are fully initialized, the jobs should transition from 'Runnable' to 'Running' in the Batch dashboard, and begin processing their task. You will occasionally see Spot EC2 instances listed as 'Terminated'; these are instances that have been reclaimed by Spot, and their jobs will be resubmitted to another instance.

The Nextflow report and timeline files are another useful source of information on the progress of the job. You will observe that occasionally, some task jobs submitted to the default queue will fail due to the underlying Spot EC2 instance being interrupted. In this case, the Nextflow report should show that the relevant job was resubmitted and ran to successful completion.