Microsoft

Module 02: Implement batch jobs by using Azure Batch services

Kishore Chowdary

Microsoft

Topics

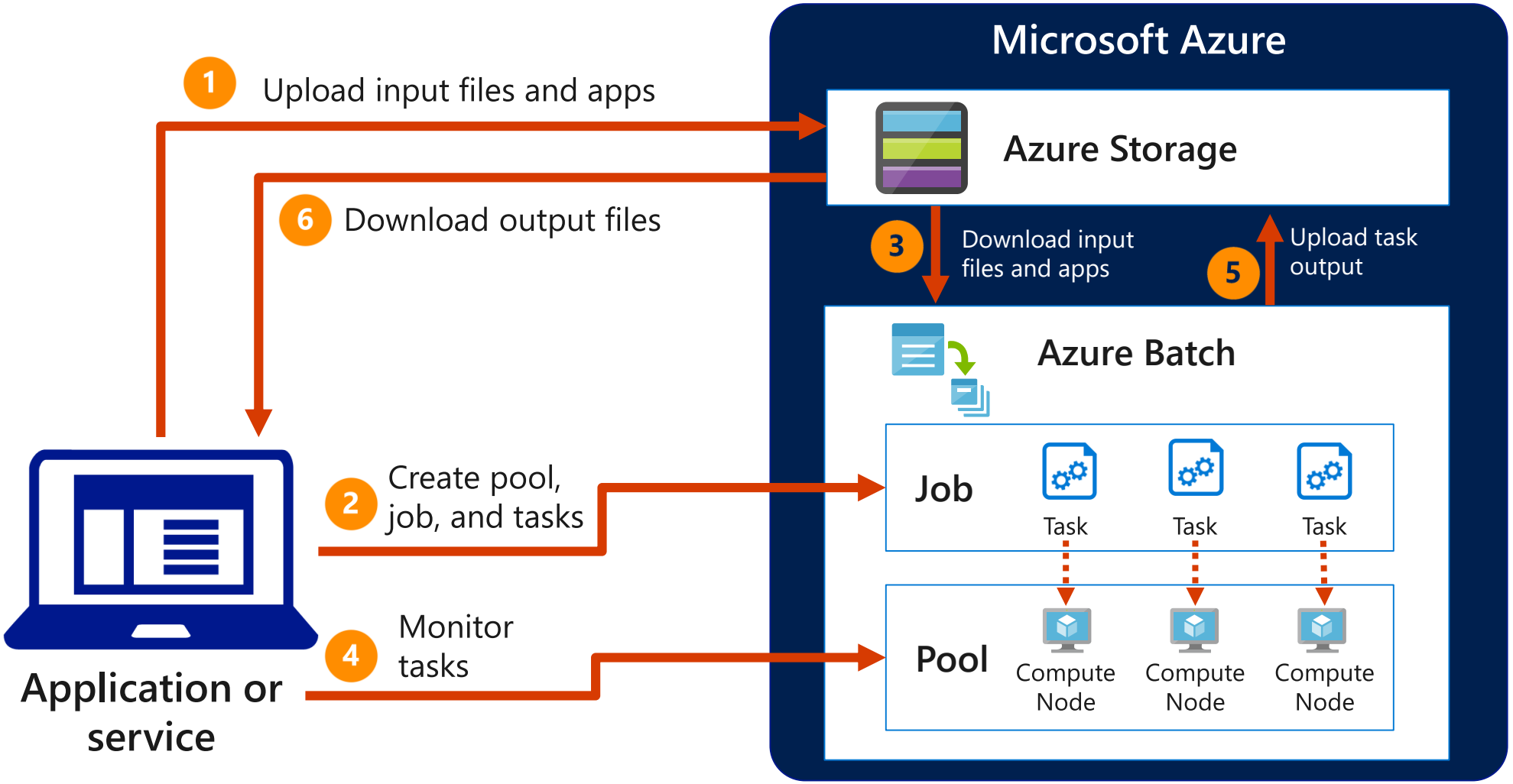
- Azure Batch
- Run a batch job by using Azure CLI and Azure portal
- Running batch jobs by using code

Lesson 01: Azure Batch

Azure Batch

- High-performance computing (HPC) describes the aggregation of complex processes across many different machines, thereby maximizing the computing power of all the machines
- Two types of workloads
  - Massively parallel
    - There needs to be a very large quantity of independent nodes processing simultaneously
  - Tightly-coupled
    - Multiple nodes need to talk often and in a very fast manner
- Service that manages VMs for large-scale parallel and HPC applications
  - Autoscaling functionality
  - Job tracking/scheduling/management functionality

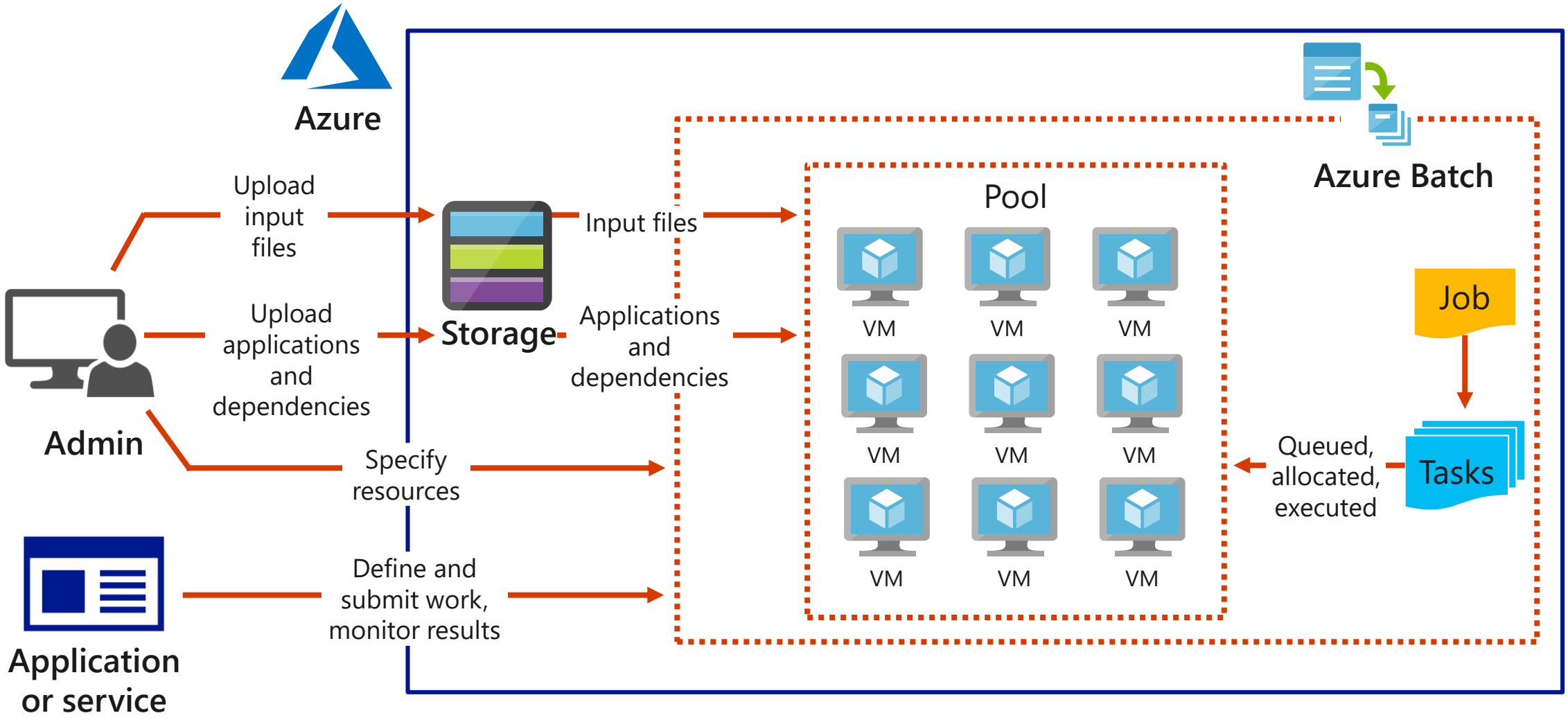
How Azure Batch works



How Azure Batch works: breakdown

1. Upload input files and the applications to process those files to your Azure Storage account
2. Create a Batch pool of compute nodes in your Batch account, a job to run the workload on the pool, and tasks in the job
3. Download input files and the applications to Batch
4. Monitor task execution
5. Upload task output
6. Download output files

Azure Batch example



Azure Batch service resources

- Account
  - Uniquely identified entity within the Batch service
- Azure Storage account
  - Stores resource files and eventually output files
- Compute node
  - A virtual machine dedicated to processing your application’s workload
- Pool
  - Collection of compute nodes that runs the entire application

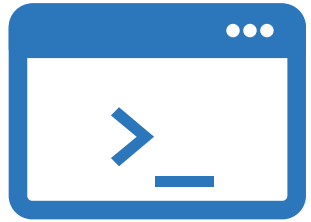
Quotas and limits

Resource	Default Limit	Maximum Limit
Batch accounts per region per subscription	1 - 3	50
Dedicated cores per Batch account	10 - 100	N/A
Low-priority cores per Batch account	10 - 100	N/A
Active jobs and job schedules per Batch account	100 - 300	1000
Pools per Batch account	20 - 100	500

Lesson 02: Run a Batch job by using Azure CLI and Azure portal

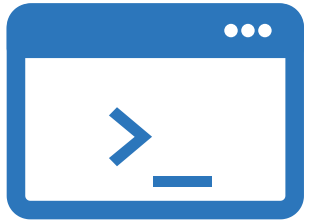
Creating Batch account with Azure CLI

```
az batch account create \
    --name mybatchaccount \
    --storage-account mystorageaccount \
    --resource-group myResourceGroup \
    --location eastus2

az batch account login \
    --name mybatchaccount \
    --resource-group myResourceGroup \
    --shared-key-auth
```



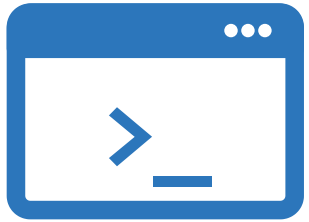
Creating Batch pools and jobs with Azure CLI

```
az batch pool create \
    --id mypool --vm-size Standard_A1_v2 \
    --target-dedicated-nodes 2 \
    --image canonical:ubuntuserver:16.04-LTS \
    --node-agent-sku-id "batch.node.ubuntu 16.04"

az batch pool show --pool-id mypool \
    --query "allocationState"

az batch job create \
    --id myjob \
    --pool-id mypool
```



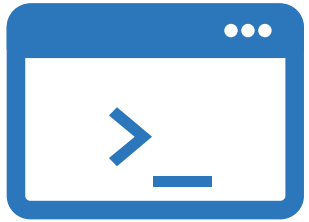
Running Batch jobs with Azure CLI

```
for i in {1..4}
do
    az batch task create \
        --task-id mytask$i \
        --job-id myjob \
        --command-line "/bin/bash -c 'printenv | grep AZ_BATCH; sleep 90s'"
done

az batch task show \
    --job-id myjob \
    --task-id mytask1
```



Viewing Batch job output with Azure CLI

```
az batch task file list \
    --job-id myjob \
    --task-id mytask1 \
    --output table

az batch task file download \
    --job-id myjob \
    --task-id mytask1 \
    --file-path stdout.txt \
    --destination ./stdout.txt
```



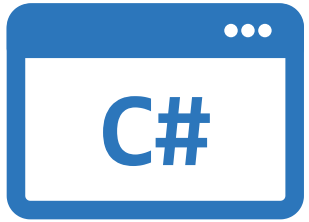
Lesson 03: Running Batch jobs by using code

Batch Management client library

- .Net library used to automate Batch account:
  - Creation
  - Deletion
  - Key management
  - Quota discovery
- You can use the library to:
  - Create and delete Batch accounts within any region
  - Retrieve and regenerate account keys programmatically
  - Check account quotas before starting jobs, creating pools, or adding compute nodes
  - Combine Batch with features of other Azure services

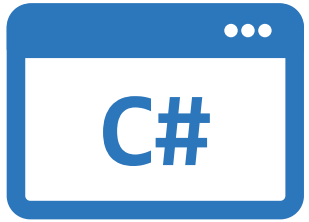
Create and delete Batch accounts

```
// Create a new Batch account
await batchManagementClient.Account.CreateAsync("MyResourceGroup",
    "mynewaccount",
    new BatchAccountCreateParameters() { Location = "West US" });

// Get the new account from the Batch service
AccountResource account = await batchManagementClient.Account.GetAsync(
    "MyResourceGroup",
    "mynewaccount");

// Delete the account
await batchManagementClient.Account.DeleteAsync("MyResourceGroup",
    account.Name);
```



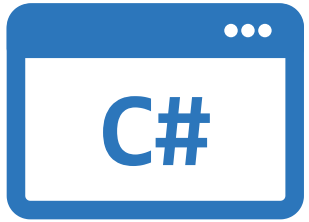
Retrieve and regenerate account keys

```
// Get and print the primary and secondary keys
BatchAccountListKeyResult accountKeys =
    await batchManagementClient.Account.ListKeysAsync("MyResourceGroup",
"mybatchaccount");
Console.WriteLine("Primary key: {0}", accountKeys.Primary);
Console.WriteLine("Secondary key: {0}", accountKeys.Secondary);

// Regenerate the primary key
BatchAccountRegenerateKeyResponse newKeys =
    await batchManagementClient.Account.RegenerateKeyAsync(
        "MyResourceGroup", "mybatchaccount",
        new BatchAccountRegenerateKeyParameters() {
            KeyName = AccountKeyType.Primary
    });
```



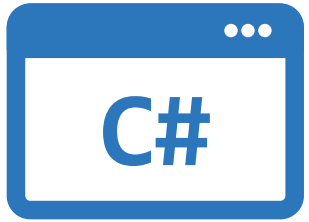
Checking Azure and Batch account quotas

```
// Get a collection of all Batch accounts within the subscription
BatchAccountListResponse listResponse =
    await batchManagementClient.Account.ListAsync(new
AccountListParameters());
IList<AccountResource> accounts = listResponse.Accounts;
Console.WriteLine("Total number of Batch accounts under subscription id {0}:
{1}",
    creds.SubscriptionId,
    accounts.Count);

// Get a count of all accounts within the target region
string region = "westus";
int accountsInRegion = accounts.Count(o => o.Location == region);
```



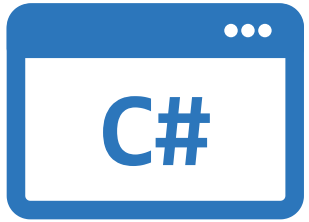
Checking Azure and Batch account quotas for regions

```
// Get the account quota for the specified region
SubscriptionQuotasGetResponse quotaResponse = await
batchManagementClient.Subscriptions.GetSubscriptionQuotasAsync(region);

Console.WriteLine("Account quota for {0} region: {1}", region,
quotaResponse.AccountQuota);

// Determine how many accounts can be created in the target region
Console.WriteLine("Accounts in {0}: {1}", region, accountsInRegion);
Console.WriteLine("You can create {0} accounts in the {1} region.",
quotaResponse.AccountQuota - accountsInRegion, region);
```



Checking Azure and Batch account resource quotas

```
// First obtain the Batch account
BatchAccountGetResponse getResponse =
    await batchManagementClient.Account.GetAsync("MyResourceGroup",
"mybatchaccount");

AccountResource account = getResponse.Resource;

// Now print the compute resource quotas for the account
Console.WriteLine("Core quota: {0}", account.Properties.CoreQuota);
Console.WriteLine("Pool quota: {0}", account.Properties.PoolQuota);
Console.WriteLine("Active job and job schedule quota: {0}",
account.Properties.ActiveJobAndJobScheduleQuota);
```



Demo: Running Batch jobs

Microsoft

Review

- Azure Batch
- Run a Batch job by using Azure CLI and Azure portal
- Running Batch jobs