



**BRENT OZAR**  
UNLIMITED®

# How to Build a Microsoft SQL Server Disaster Recovery Plan with Google Compute Engine



Author: Tara Kizer

Technical Reviewer: Brent Ozar

© 2017 Brent Ozar Unlimited.

Version: v1.0, 2017-03-03. To get the latest version free, visit: <https://BrentOzar.com/go/gce>

# Table of Contents

## [Table of Contents](#)

### [Introduction: About You, Us, and This Build](#)

[About You](#)

[About Us](#)

[About What We'll Build Together](#)

[About the Licensing](#)

### [Creating a Storage Bucket to Hold Our Backups](#)

### [Copying SQL Server Backups to the Cloud](#)

[Shipping Backups from SQL Server](#)

[Downloading and Installing the Google Cloud SDK](#)

[Synchronizing the Backup Folders with Rsync](#)

[Automating Rsync with a SQL Server Agent Job](#)

[Shipping Backups from a Backup Share Server](#)

[Automating Rsync with Task Scheduler](#)

[Next Steps Before Going Into Production](#)

### [Failing Over to GCE](#)

[Creating a VM Using the UI](#)

[Creating a VM Using the Command Line](#)

[Connecting to the VM](#)

[Configuring the VM](#)

[Restoring the Database](#)

### [Failing Back to the Primary Server](#)

### [Recap and What to Consider Next](#)

# Introduction: About You, Us, and This Build

## About You

You're a systems administrator or database administrator who wants to protect your production SQL Server. However, you don't have a separate data center or another colo site.

You're looking for instructions on:

- How to copy your SQL Server databases to Google Cloud Storage
- How to spin up a SQL Server VM in Google Compute Engine to test your backups
- How to use our free scripts to restore your backups to the most recent point in time
- How to test the backups to make sure they're corruption-free

Your goal here is a very, very inexpensive disaster recovery option. It won't be a hot standby ready to automatically fail over - there's going to be manual labor involved - but at least you won't be starting from scratch.

## About Us

I'm your tour guide, Tara Kizer, and I'll be assisted by Brent Ozar, who will be your technical reviewer for this white paper.

We're from Brent Ozar Unlimited®, a small boutique consulting firm that makes SQL Server faster and more reliable. You can find more of our work at <https://www.BrentOzar.com>.

## About What We'll Build Together

We like to call this Log Shipping 2.0.

With conventional database transaction log shipping, every time the primary SQL Server takes a log backup, another SQL Server notices the newly created backup file and restores it. This way, you've always got a ready-to-go standby SQL Server.

Thing is, that costs money.

We're going to take the cloud approach. We're going to still take log backups every X minutes (with X being a number we'll discuss), then sync those files up to the cloud. Then, when disaster strikes (or you want to just test your backups), we'll spin up a SQL Server in the cloud, restore those backups, and make sure our databases are free from corruption.

Believe it or not, the hardest thing about what we're about to build is just syncing the backup files up into the cloud. There's a lot of gotchas to think about:

- These files may not be small (especially your full backups)
- The files may not be stored on your SQL Server (like you might be writing your backups to a file share or UNC path)
- Your Internet bandwidth may be limited
- We're going to rely on command-line tools for the syncing

As long as you understand the concepts, you could totally build this same solution with any chosen file-sync tool.

Assumptions:

- We've already got a production SQL Server up and running.
- We're writing our backups locally to the D: drive. (This isn't a best practice - we'd rather [write our backups to a network share](#), but we're going to keep this white paper simple.)
- We're using Windows Server 2016 and SQL Server 2016 for this project, but almost everything we'll show should work highly similarly on previous versions. (Test first, obviously - you don't wanna be learning this stuff when disaster strikes.)

## About the Licensing

High availability and disaster recovery licensing gets really tricky for SQL Server, especially when you're spanning between on-premises servers and cloud servers.

Before SQL Server 2012, you basically got one free standby SQL Server for every active primary server that you licensed. This meant that you could leave that standby up and running, constantly restoring your transaction log backups, at a low cost.

Unfortunately, starting with SQL Server 2012, you need to be licensed with Software Assurance in order to get the "free" standby server. Software Assurance is the ongoing maintenance fee for SQL Server that gets you free upgrades, plus a few other benefits (like virtualization mobility and this passive standby thing.)

To learn more about SQL Server licensing, start here:

- [SQL Server Licensing Simplified Into 7 Rules](#)
- [How to Ask a SQL Server Licensing Question](#)

Having said that, in Google Compute Engine, there are two ways to license your SQL Server.

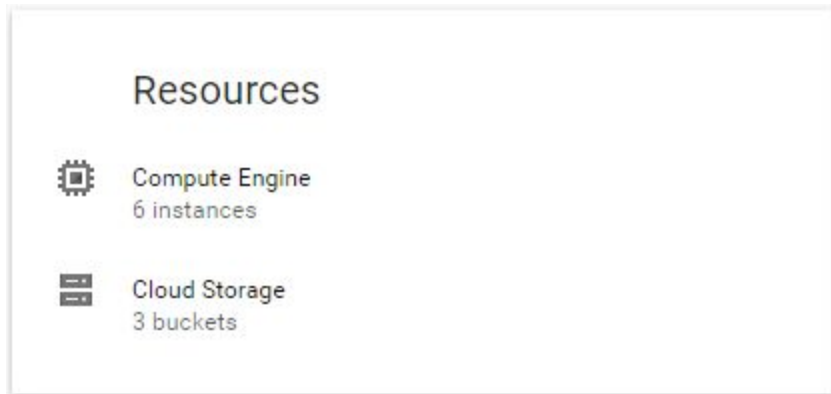
**With pay-per-use licensing**, your GCE VM hourly cost includes licensing. Google manages the licensing logistics with Microsoft. Your hourly costs are higher, but you have total flexibility to ramp your costs up and down whenever you want.

**With bring-your-own-licensing (BYOL)**, your GCE VM costs are lower because the licensing isn't included. You'll need to purchase your own SQL Server licensing from Microsoft (like using your passive-standby benefit from Software Assurance), which means paying up front, and you don't have much flexibility here. However, for companies with very stable usage needs, or with free/discounted licensing through Microsoft licensing agreements, this can end up cheaper. If you're installing Developer Edition, or if you're very confident in your Software Assurance benefits, use this approach.

With this particular project, we're going to minimize licensing by only spinning a server up every now and then to test our backups. To play it on the safe side, consider pay-per-use licensing here.

# Creating a Storage Bucket to Hold Our Backups

Create a storage bucket in GCE by clicking on *Cloud Storage* under *Resources* in the GCE dashboard. This is like a file share in the cloud.



Click *CREATE BUCKET*, give it a name, select the storage class and regional location. Then click *Create*.

**Name** ?

Must be unique across Cloud Storage. Privacy: Do not include sensitive information in your bucket name. Others can discover your bucket name if it matches a name they're trying to use.

**Default storage class** ?[Learn about pricing](#)☐ **Multi-Regional**

Use to stream videos and host hot web content.  
Best for data accessed frequently around the world.

☒ **Regional**

Use to store data and run data analytics.  
Best for data accessed frequently in one part of the world.

☐ **Nearline**

Use to store rarely accessed documents.  
Best for data accessed less than once per month.

☐ **Coldline**

Use to store very rarely accessed documents.  
Best for data accessed less than once per year.

**Regional location**

Redundant within a single region.

**Create**

Cancel

To pick a regional location, think about what you're protecting.

If you're protecting an on-premises server, then GCE is probably your disaster recovery failover site. Consider picking a region close enough to your server rack, but far enough away that you're not worried about a single disaster knocking out both your servers and GCE.

If you're protecting a server hosted in the cloud, then don't pick the same region where your primary SQL Server lives. After all, we're aiming for disaster recovery here: we want this file share to be available in the unthinkable, impossible, completely never-gonna-happen event\* that an entire GCE region goes dark.

\* This could totally happen.

# Copying SQL Server Backups to the Cloud

We will use gsutil to access Google Cloud Storage. gsutil comes with the Google Cloud SDK. If your primary server is in GCE, you already have it installed, so skip to the next section if that's the case.

You need to install the Google Cloud SDK on the machine that hosts your SQL Server's backup files. In this white paper's first example, we're doing our backups to the D: drive right on the SQL Server itself. However, if you're backing up to a network share (which is awesome!) like our second example, then you'll want to install the SDK on the file server itself so that it can manage the file synchronization process without bothering SQL Server.

If you're just doing this for the first time, do it on your desktop with a local development SQL Server instance. (Don't go installing stuff on your primary file server without knowing how it works first.)

## Shipping Backups from SQL Server

This section covers how to ship the backup files from the SQL Server to the cloud, but also use this section for how to install and configure the Google Cloud SDK if you are shipping the backup files from a backup share server to the cloud.

## Downloading and Installing the Google Cloud SDK

Download Google Cloud SDK from [here](#) and then launch the installer on the primary server.

Click *Next*.

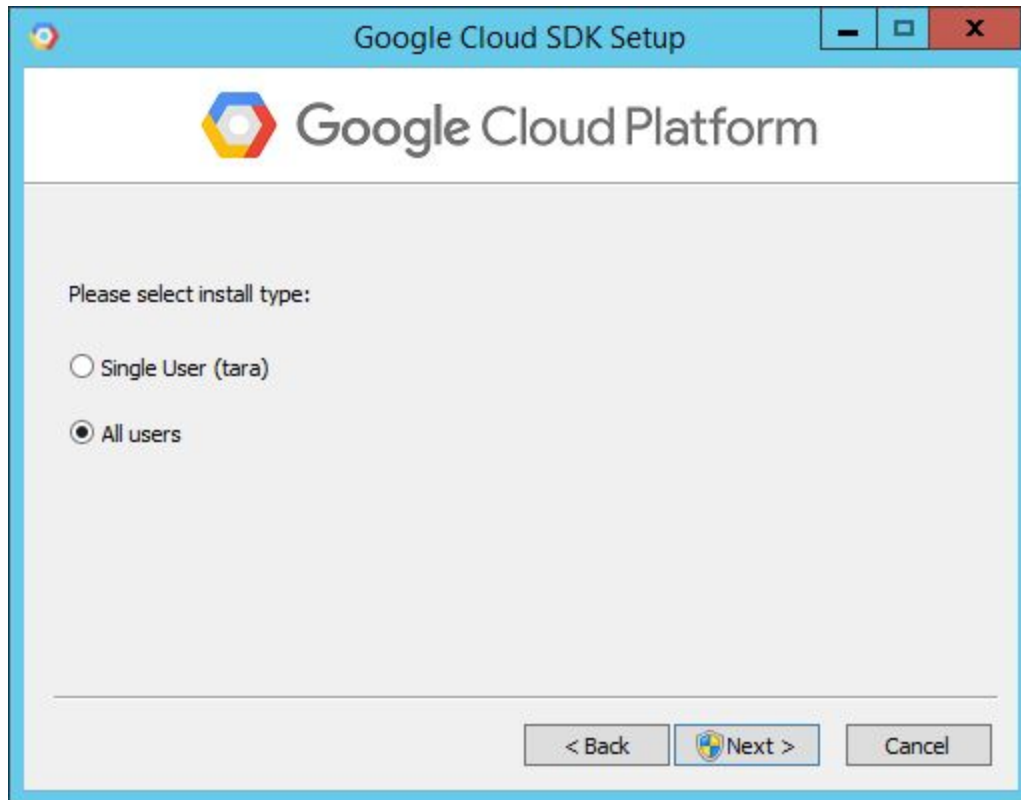




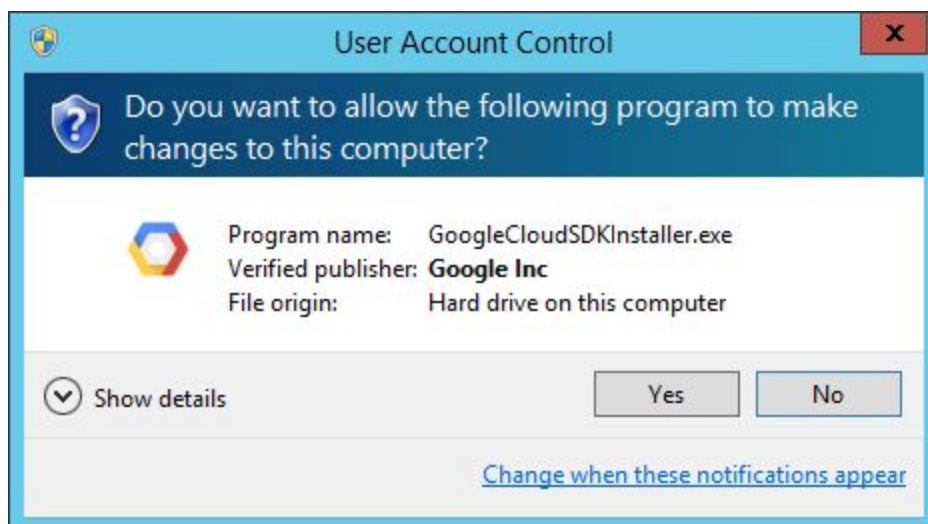
Click / Agree after you have pretended to read through the agreement.



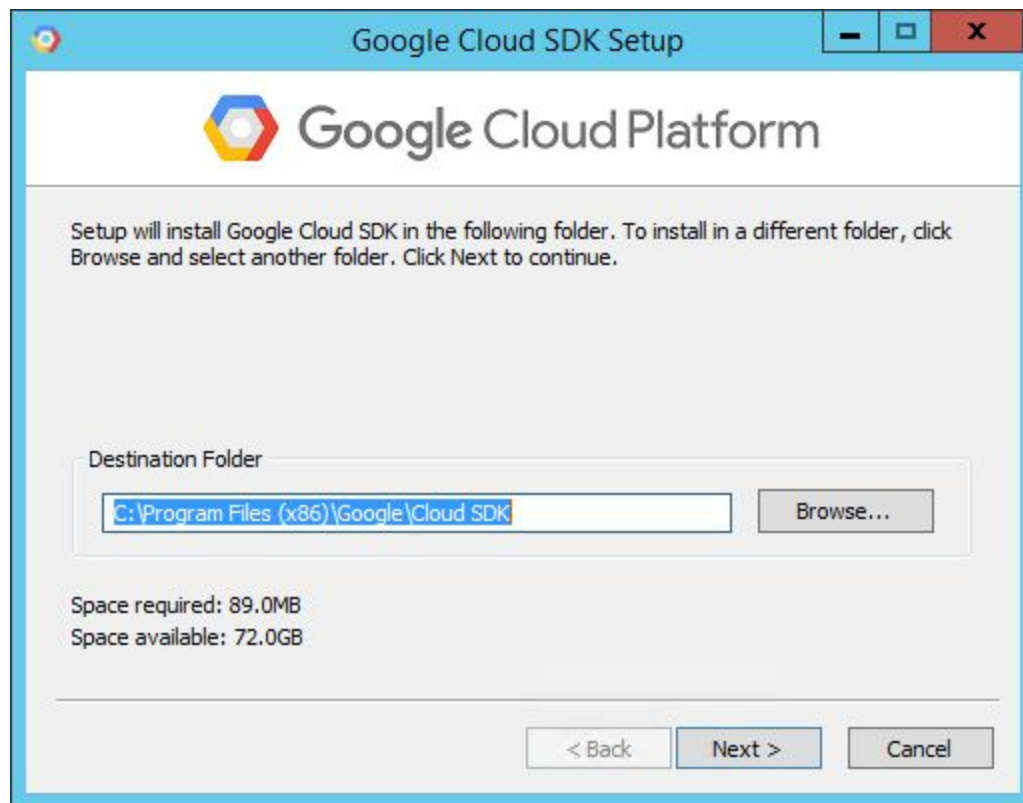
Select *All users* and then click *Next*.



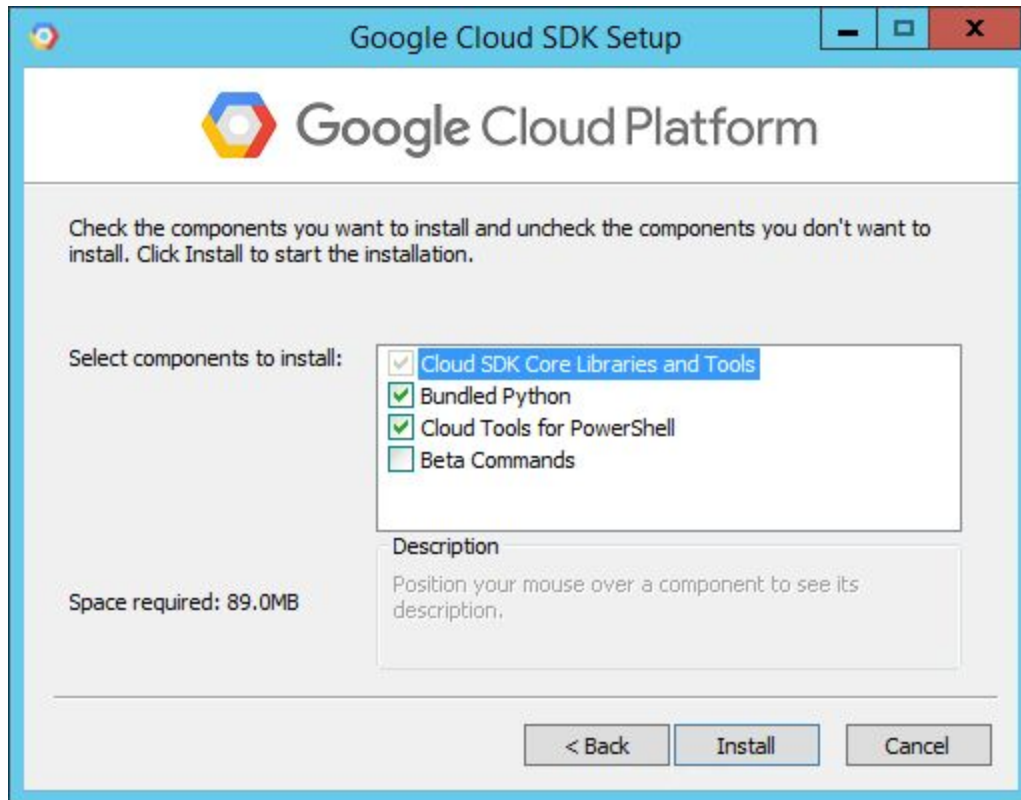
If Windows asks you to authorize to make changes, click Yes.



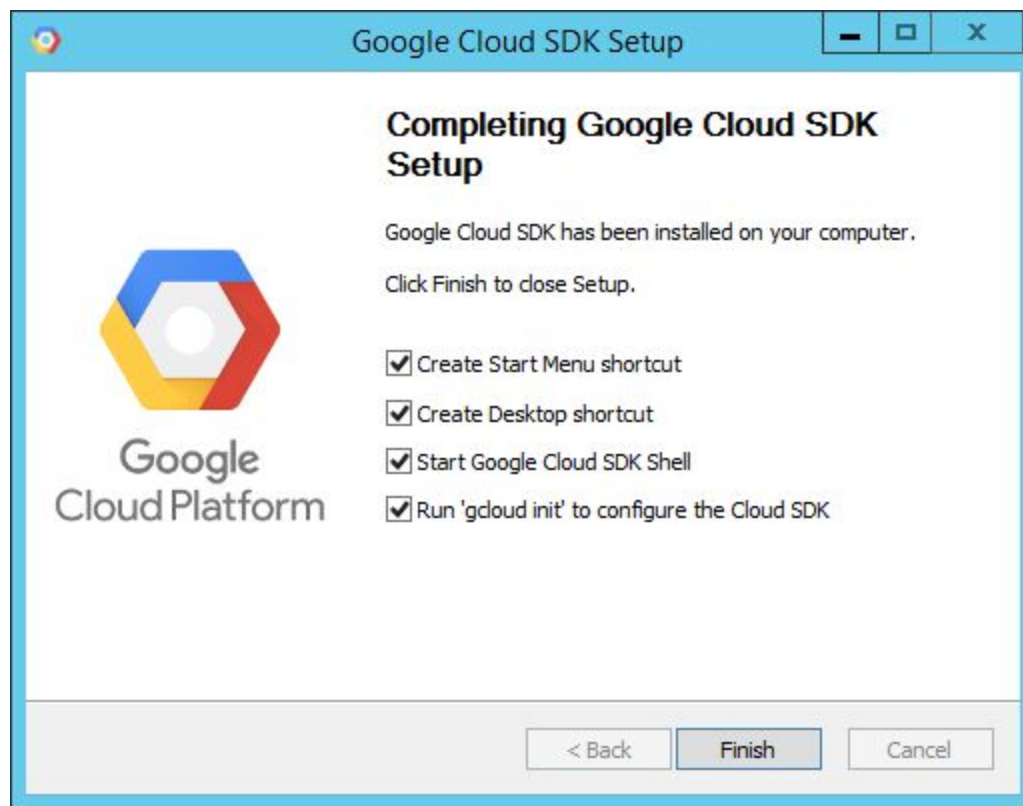
Change the path if desired and then click *Next*.



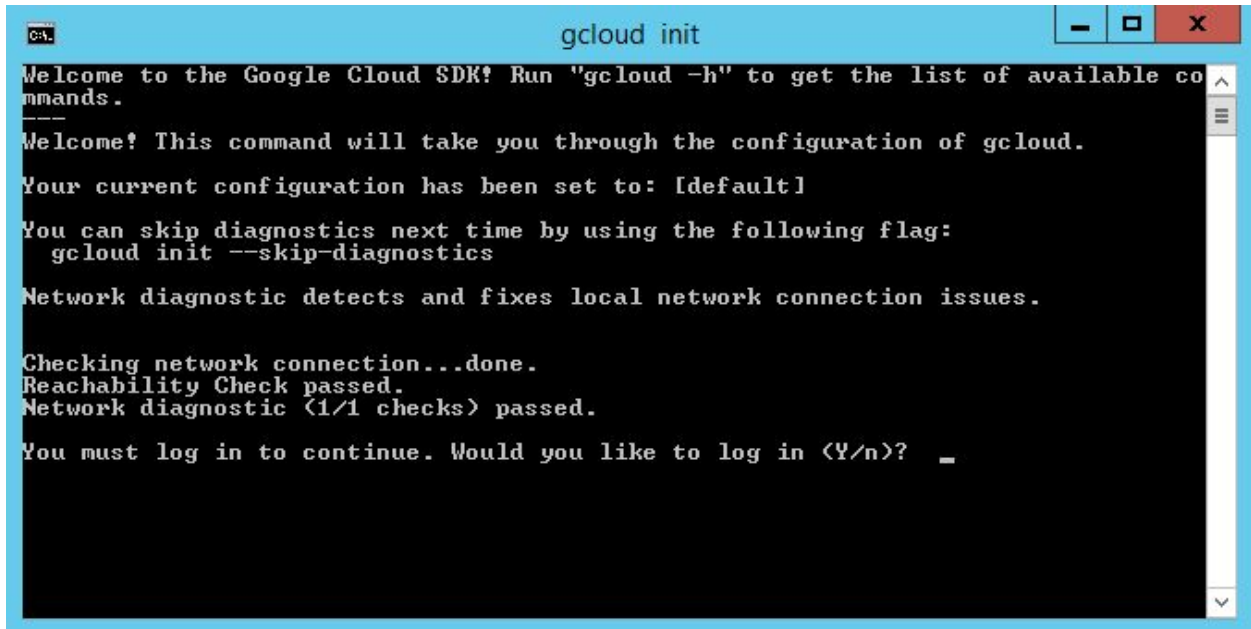
If you've already installed Python, you can unselect it. Click *Install*.



After it is successfully installed, click *Next* and then *Finish*, keeping the last two options checked so that we can easily start using gsutil.



A terminal window will appear and asks you to login. Type in Y and then hit enter. Sign into your Google account and grant it access. I had to disable Enhanced Security Configuration in Server Manager in order to use Internet Explorer (don't judge me, it's what's on the server).



```
gcloud init

Welcome to the Google Cloud SDK! Run "gcloud -h" to get the list of available commands.

Welcome! This command will take you through the configuration of gcloud.

Your current configuration has been set to: [default]

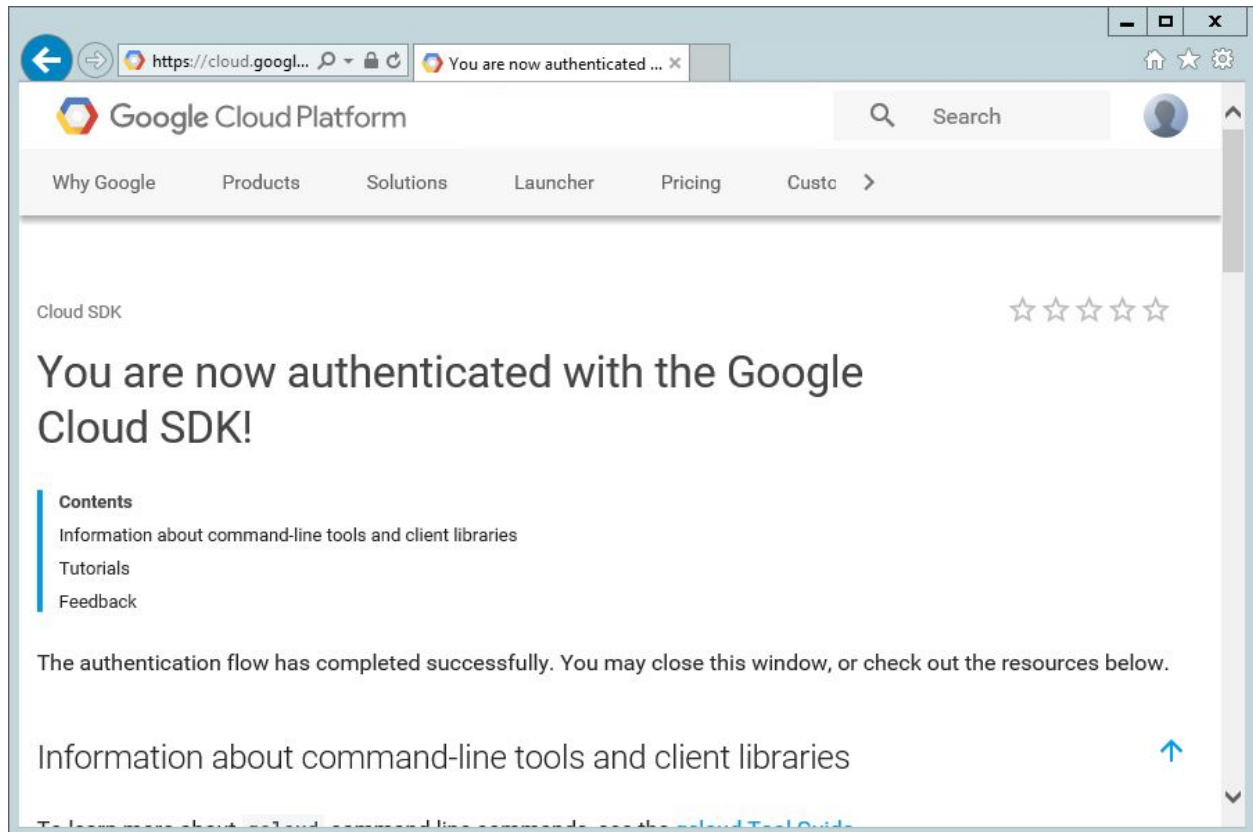
You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.

Checking network connection...done.
Reachability Check passed.
Network diagnostic (1/1 checks) passed.

You must log in to continue. Would you like to log in (Y/n)? _
```

Once you complete the login process, you are now authenticated and the command line tool is notified.



```
You are logged in as: [tara@brentozar.com].
Pick cloud project to use:
[1] always-on-availability-group
[2] sqlenterpriseimages
Please enter numeric choice or text value (must exactly match list
item):
```

Type in the name of your project and then hit *Enter*.

```
You are logged in as: [tara@brentozar.com].
Pick cloud project to use:
[1] always-on-availability-group
[2] sqlenterpriseimages
Please enter numeric choice or text value (must exactly match list
item): always-on-availability-group
```

Type Y to configure GCE.

Type in your zone's numeric choice value or 19 if you don't want to set a default zone.



```
Administrator: Command Prompt - gcloud init
Which Google Compute Engine zone would you like to use as project
default?
If you do not specify a zone via a command line flag while working
with Compute Engine resources, the default is assumed.
[1] asia-east1-a
[2] asia-east1-b
[3] asia-east1-c
[4] asia-northeast1-a
[5] asia-northeast1-b
[6] asia-northeast1-c
[7] europe-west1-c
[8] europe-west1-b
[9] europe-west1-d
[10] us-central1-f
[11] us-central1-b
[12] us-central1-c
[13] us-central1-a
[14] us-east1-d
[15] us-east1-c
[16] us-east1-b
[17] us-west1-b
[18] us-west1-a
[19] Do not set default zone
Please enter numeric choice or text value <must exactly match list
item>: _
```

I chose not to set it. In the next screen, type in your region's numeric choice value or 7 if you don't want to set a default region.

```
Which Google Compute Engine region would you like to use as project
default?
If you do not specify a region via a command line flag while working
with Compute Engine resources, the default is assumed.
[1] asia-east1
[2] asia-northeast1
[3] europe-west1
[4] us-central1
[5] us-east1
[6] us-west1
[7] Do not set default region
Please enter numeric choice or text value <must exactly match list
item>: _
```

I chose not to set it. You are now ready to use gsutil - or at least, gsutil is ready for you. You might still need a little book-reading first.

```
Administrator: Command Prompt

Please enter numeric choice or text value (must exactly match list item): 7

Created a default .boto configuration file at [C:\Users\tara\.boto]. See this file and
[https://cloud.google.com/storage/docs/gsutil/commands/config] for more
information about configuring Google Cloud Storage.
Your Google Cloud SDK is configured and ready to use!

* Commands that require authentication will use tara@brentozar.com by default
* Commands will reference project 'always-on-availability-group' by default
Run 'gcloud help config' to learn how to change individual settings

This gcloud configuration is called [default]. You can create additional configurations if you work with multiple accounts and/or projects.
Run 'gcloud topic configurations' to learn more.

Some things to try next:

* Run 'gcloud --help' to see the Cloud Platform services you can interact with.
And run 'gcloud help COMMAND' to get help on any gcloud command.
* Run 'gcloud topic -h' to learn about advanced features of the SDK like arg files and output formatting

C:\Windows\system32>
```

To see a listing of the files in the storage bucket, use *gsutil ls*.

## Synchronizing the Backup Folders with Rsync

```
C:\Windows\system32>gsutil ls gs://log_shipping_backup/LOG_
```

To copy the contents of a folder, use *gsutil rsync*. ([Rsync](#) is a utility for keeping files in sync across different systems. It's not exactly known for its ease of use or bulletproof reliability, but it works well enough for our evil purposes here.)


```
C:\Windows\system32>gsutil rsync -d -r C:\Temp\LOG gs://log_shipping_backup/LOG/LOG
Building synchronization state...
Starting synchronization
Copying file://C:\Temp\LOG\New Text Document.txt [Content-Type=text/plain]...
/ [0 files] 0.0 B/ 0.0 B]
/ [1 files] 0.0 B/ 0.0 B]


Operation completed over 1 objects.

C:\Windows\system32>
```


Now time to automate it. We need to be able to push files without manually having to authenticate, so we need to setup a service account to authenticate.


In the GCE dashboard, click *Manage project settings* in the project box.


 **Project: Always-On-Availability-Group**  
ID: always-on-availability-group (# 159779191381)


 [Manage project settings](#)


Click *Service accounts*.


 **IAM & Admin**


 All projects


 IAM

 Quotas

 **Service accounts**

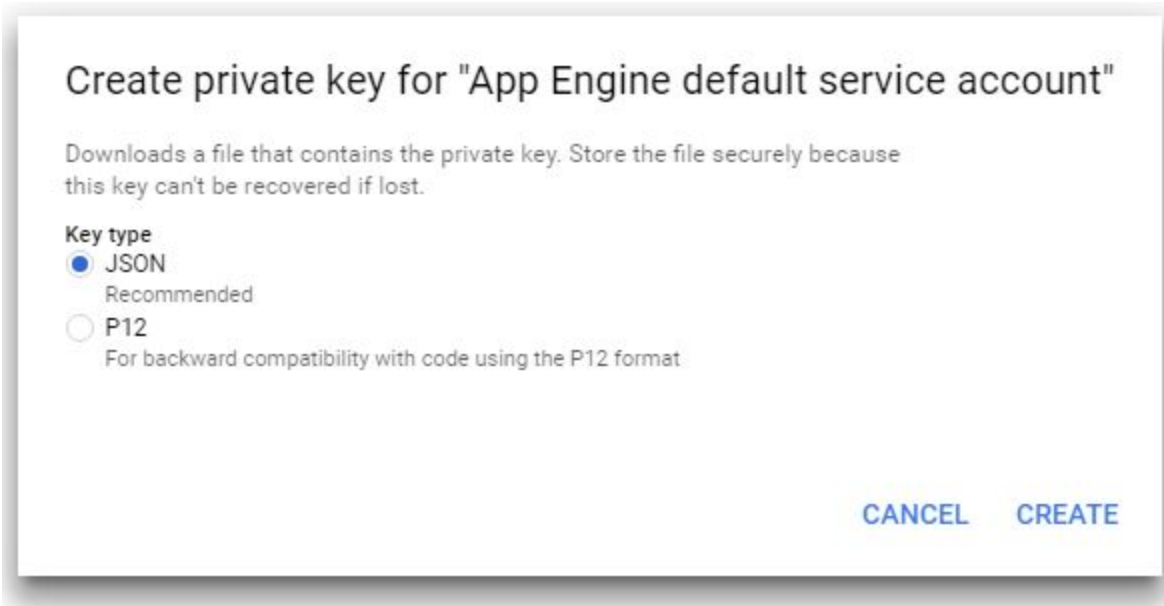
 Labels

 GCP Privacy & Security

 Settings

You can create a new service account or use an existing one. Over on the right for the service account, select *Create key* after clicking on the dots.

Use *JSON* for the *Key type* and then click *CREATE*.



The file is now downloaded to the computer where you are accessing the GCE console. Copy the file to the primary server and then run `gcloud auth activate-service-account`, passing in the service account and JSON file.

```
gcloud auth activate-service-account
always-on-availability-group@appspot.gserviceaccount.com --key-file
C:\Temp\Always-On-Availability-Group-0e2476b69c5e.json
```

```
C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin>gcloud auth activate-service-account always-on-availability-group@appspot.gserviceaccount.com --key-file C:\Temp\Always-On-Availability-Group-0e2476b69c5e.json
Activated service account credentials for: [always-on-availability-group@appspot.gserviceaccount.com]
```

Run `gsutil rsync` to create the folders and copy the initial set of files.

```
gsutil rsync -d -r D:\MSSQL\Backup gs://log_shipping
```

```
Administrator: Command Prompt
C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin>gsutil rsync -d -r
D:\MSSQL\Backup gs://log_shipping/
Building synchronization state...
Starting synchronization
Copying file:///D:\MSSQL\Backup\SQL2016PROD1A\LogShipMe\FULL\SQL2016PROD1A_LogShip
pMe_FULL_20170112_205214.bak [Content-Type=application/octet-stream]...
\ [0 files] 0.0 MiB/ 2.7 MiB
\ [1 files] 2.7 MiB/ 2.7 MiB
Copying file:///D:\MSSQL\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShip
Me_LOG_20170112_205224.trn [Content-Type=application/octet-stream]...
\ [1 files] 2.7 MiB/ 2.8 MiB
\ [2 files] 2.8 MiB/ 2.8 MiB
Copying file:///D:\MSSQL\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShip
Me_LOG_20170112_205500.trn [Content-Type=application/octet-stream]...
\ [2 files] 2.8 MiB/ 2.9 MiB
\ [3 files] 2.9 MiB/ 2.9 MiB
Copying file:///D:\MSSQL\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShip
Me_LOG_20170112_210000.trn [Content-Type=application/octet-stream]...
\ [3 files] 2.9 MiB/ 3.0 MiB
\ [4 files] 3.0 MiB/ 3.0 MiB

==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m -o ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.

Copying file:///D:\MSSQL\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShip
Me_LOG_20170112_210500.trn [Content-Type=application/octet-stream]...
\ [4 files] 3.0 MiB/ 3.0 MiB
\ [5 files] 3.0 MiB/ 3.0 MiB
Copying file:///D:\MSSQL\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShip
Me_LOG_20170112_211000.trn [Content-Type=application/octet-stream]...
```

## Automating Rsync with a SQL Server Agent Job

Create a job to push the LOG backups to the storage bucket.

Job Properties - Sync Backups to the Cloud

Select a page

- General
- Steps
- Schedules
- Alerts
- Notifications
- Targets

Script Help

Name: Sync LOG Backups to the Cloud

Owner: sa

Category: [Uncategorized (Local)]

Description: No description available.

Connection

Server: SQL2016PROD1A

Connection: lab\tara

[View connection properties](#)

Progress

Ready

☒ Enabled

Source:

Created: 1/12/2017 7:18:14 PM

Last modified: 1/12/2017 8:56:25 PM

Last executed: 1/12/2017 8:44:31 PM

[View Job History](#)

OK Cancel

For the job step, use *Operating system (CmdExec)* as the *Type*. Add the gsutil command and then click *OK*.

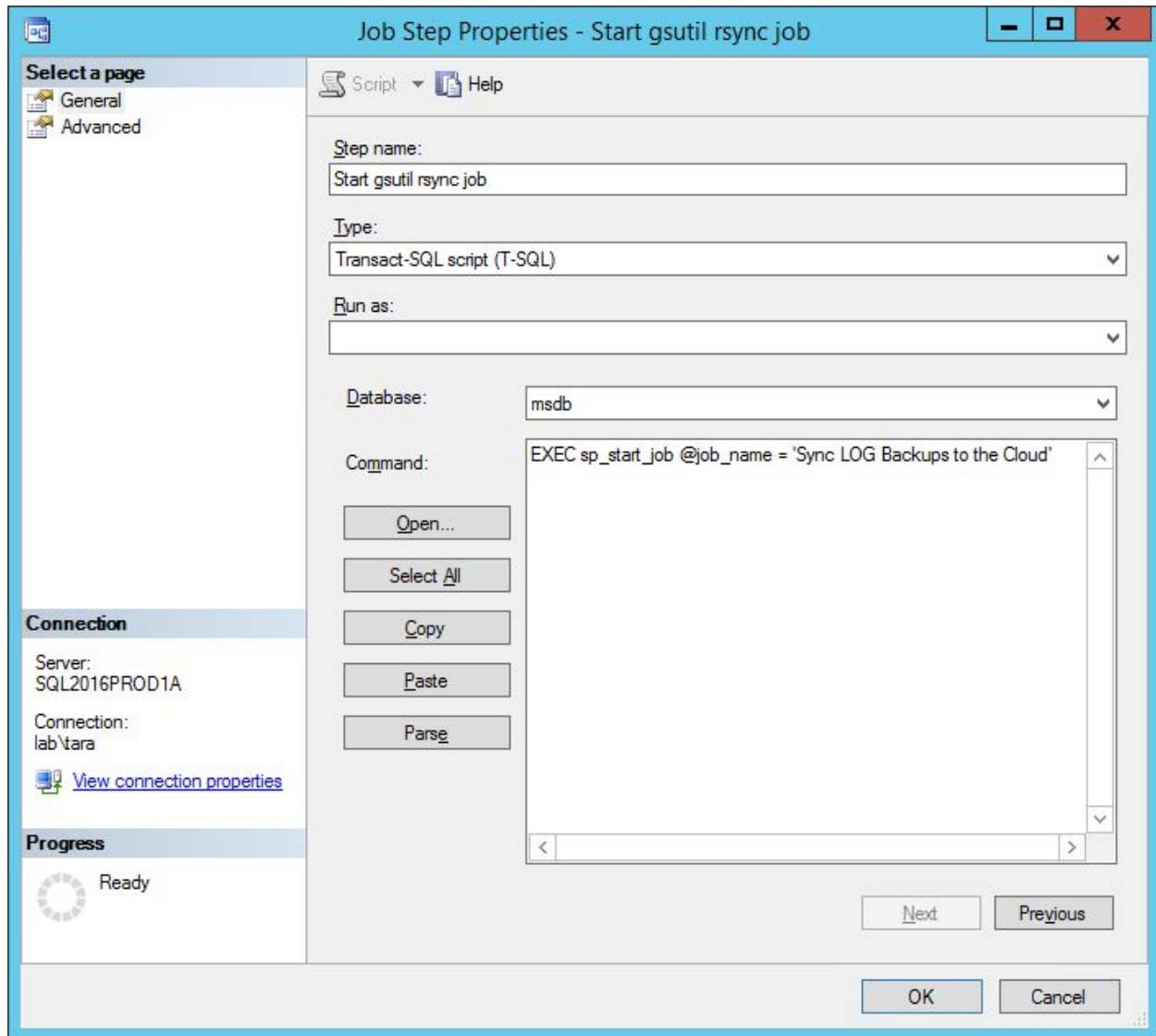
```
"C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\gsutil.cmd" rsync -d -r
D:\MSSQL\Backup\SQL2016PROD1A\LogShipMe\LOG\
gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/
```

Click *OK* again to create the job. Note that I didn't create a schedule. As the *gsutil* command will fail if there are no files to copy over, it should be ran after the LOG backup job completes.

Add a job step to the LOG backup job and add an *sp\_start\_job* command to kickoff the job that was just created.

```
EXEC sp_start_job @job_name = 'Sync LOG Backups to the Cloud'
```





On the *Advanced* page, change the *On success action* to *Quit the job reporting success* and then click *OK*. Modify the first job step's *On success action* to *Go to the next step* and then click *OK*. Click *OK* to complete modifying the LOG backup job.

Repeat this process for the FULL backups and DIFF backups if applicable.

1. Create a job to run the *gsutil rsync* command with no schedule
2. Modify the backup job to add a new step to kickoff the just-created job
3. Modify the *On success action* for both job steps.

Alternatively, you could add the *gsutil* command directly to the LOG backup job, but I prefer to separate my jobs for ease of running them independently.



Test out the backup jobs to ensure everything is working properly. You can use *gsutil ls* to see which files and folders are now in the cloud.

```
gsutil ls gs://log_shipping/SQL2016PROD1A/LogShipMe/FULL
```

```
gsutil ls gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG
```

Now let's look at setting this up when the backups aren't stored locally but rather the backup jobs are pointed at a network share.

## Shipping Backups from a Backup Share Server

RDP to the backup share server and login with a service account.

Install Google Cloud SDK on the server where the backup share exists. Usually this is some kind of file server that doesn't have SQL Server installed.

You won't need to configure gcloud again as that's at the account level and not the server level. It was already configured from the work that was done on the primary server.

Test that gsutil is working properly.

```
gsutil ls gs://log_shipping
```

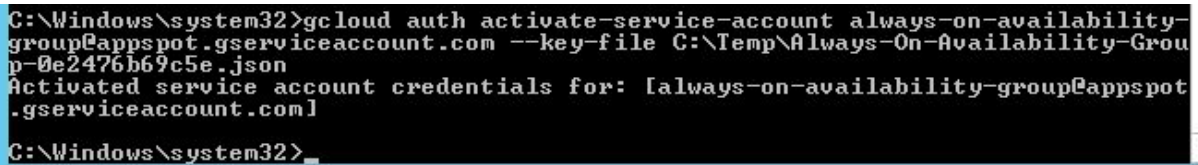


```
C:\Windows\system32>gsutil ls gs://log_shipping
gs://log_shipping/SQL2016PROD1A/
C:\Windows\system32>
```

## Automating Rsync with Task Scheduler

Say you're backing up to a network share hosted on a Windows box instead of locally on the SQL Server. Either run through the above setup steps on the Windows box, or copy the JSON file from the primary server to the backup share server and then run the `gcloud auth activate-service-account`.

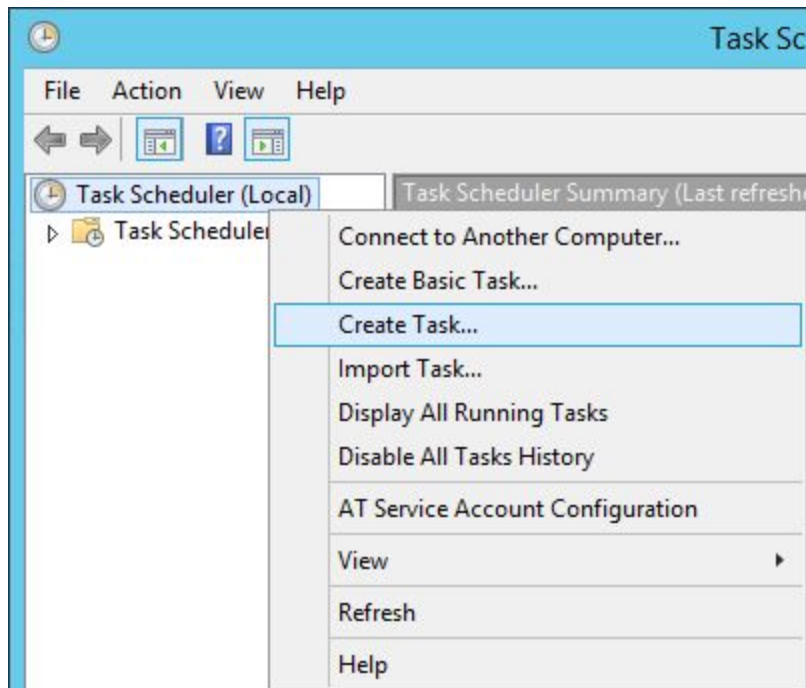
```
gcloud auth activate-service-account  
always-on-availability-group@appspot.gserviceaccount.com --key-file  
C:\Temp\Always-On-Availability-Group-0e2476b69c5e.json
```



```
C:\Windows\system32>gcloud auth activate-service-account always-on-availability-  
group@appspot.gserviceaccount.com --key-file C:\Temp\Always-On-Availability-Grou  
p-0e2476b69c5e.json  
Activated service account credentials for: [always-on-availability-group@appspot  
.gserviceaccount.com]  
C:\Windows\system32>
```

Create a job to run `gsutil rsync` on a schedule. This time I'm going to use Task Scheduler since the backup share server might not have SQL Server installed.

Right click on *Task Scheduler* and select *Create Task*.



Give the job a name, have it use a service account that doesn't need to be logged in and have it run with the highest privileges.

**Create Task**

General Triggers Actions Conditions Settings

Name: Sync LOG Backups to the Cloud

Location: \

Author: lab\tara

Description:

Security options

When running the task, use the following user account:

LAB\svcSQL2016 [Change User or Group...](#)

☐ Run only when user is logged on

☒ Run whether user is logged on or not

☐ Do not store password. The task will only have access to local computer resources.

☒ Run with highest privileges

☐ Hidden

Configure for: Windows Vista™, Windows Server™ 2008

OK Cancel

On the *Triggers* tab, click *New*.

The screenshot shows a 'Create Task' dialog box with a blue title bar and a red close button. It has five tabs: 'General', 'Triggers', 'Actions', 'Conditions', and 'Settings'. The 'Triggers' tab is selected. Below the tabs is a text box with the instruction: 'When you create a task, you can specify the conditions that will trigger the task.' Below this is a table with three columns: 'Trigger', 'Details', and 'Status'. The table is currently empty. At the bottom of the table area are three buttons: 'New...', 'Edit...', and 'Delete'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

| Trigger | Details | Status |
|---------|---------|--------|
|---------|---------|--------|

Set it to run with the same frequency as the backup job that runs *Indefinitely* and then click *OK*.

**New Trigger** [X]

Begin the task: On a schedule ▼

**Settings**

☐ One time    **Start:** 1/13/2017 [calendar icon] 9:01:14 PM [time picker] ☐ Synchronize across time zones

☒ Daily    **Recur every:** 1 days

☐ Weekly

☐ Monthly

**Advanced settings**

☐ Delay task for up to (random delay): 1 hour ▼

☒ Repeat task every: 5 minutes ▼ for a duration of: Indefinitely ▼

☐ Stop all running tasks at end of repetition duration

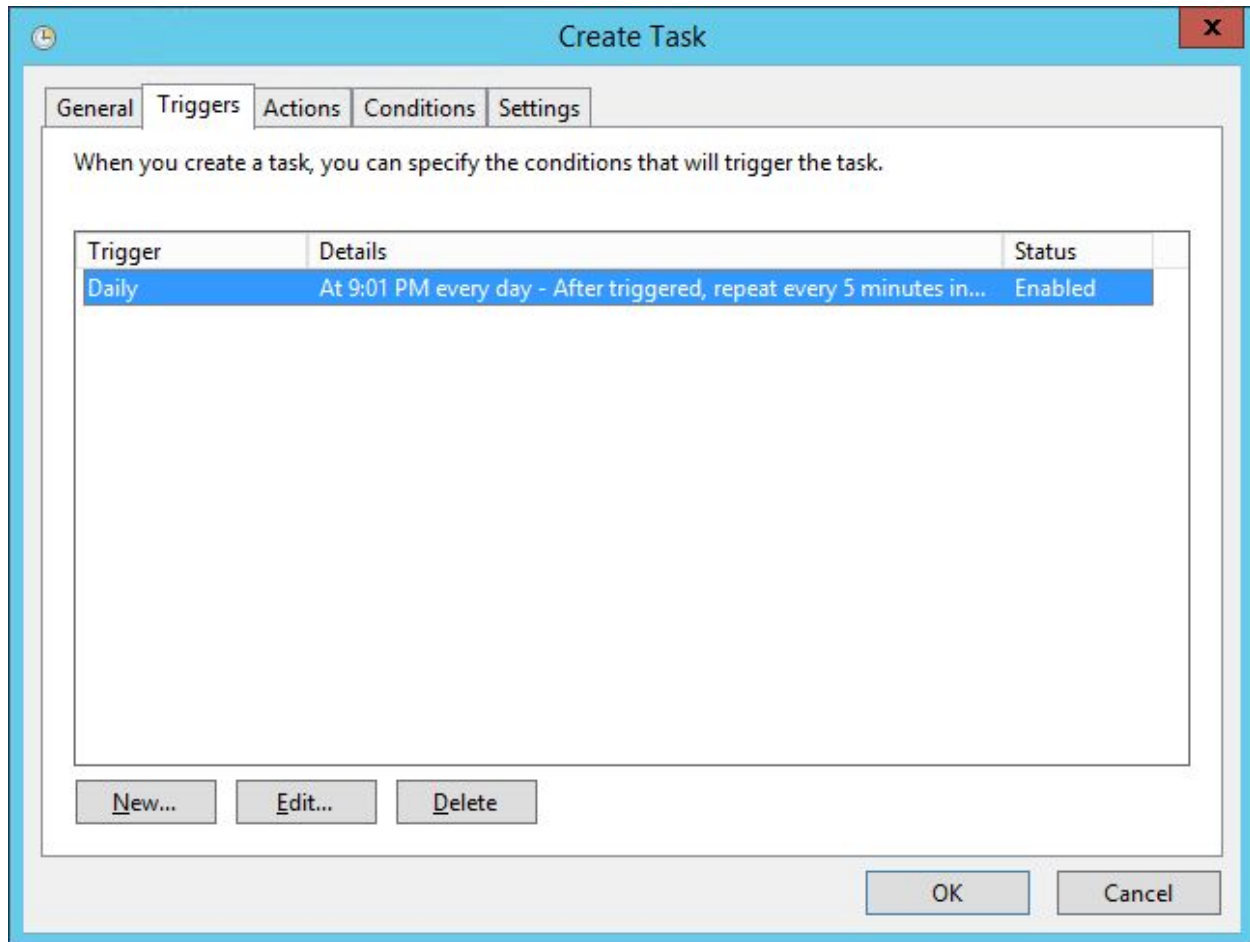
☐ Stop task if it runs longer than: 3 days ▼

☐ **Expire:** 1/13/2018 [calendar icon] 9:01:14 PM [time picker] ☐ Synchronize across time zones

☒ Enabled

OK Cancel

The *Details* column is a little confusing as it shows that it'll run at 9:01pm, but it's really going to run every 5 minutes as long as it's past 1/13/2017 at 9:01pm.



On the *Actions* tab, click *New*. Put *gsutil.cmd* and its path into the *Program/script* field and then the rest of the command into the *arguments* field. Copy the path without the double quotes, without the file name and without the final backslash (it was very picky) into the *Start in* field and then click *OK*.

*Program/script*: "C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\gsutil.cmd"

*Arguments*: rsync -d -r D:\Backups\SQL2016PROD1A\LogShipMe\LOG\  
gs://log\_shipping/SQL2016PROD1A\LogShipMe/LOG/

*Start in*: C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin

**Edit Action** X

You must specify what action this task will perform.

Action: Start a program ▾

Settings

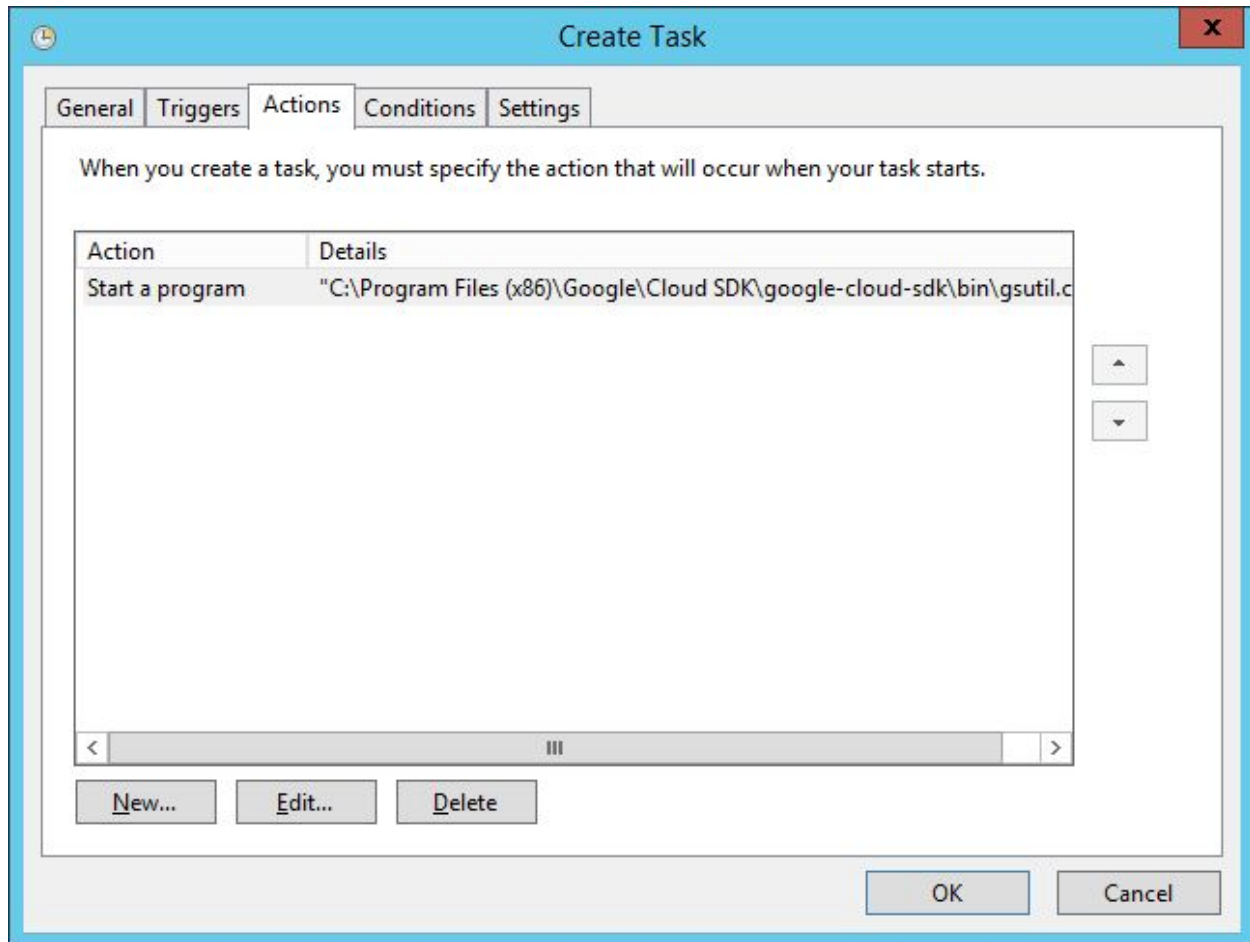
Program/script:  Browse...

Add arguments (optional):

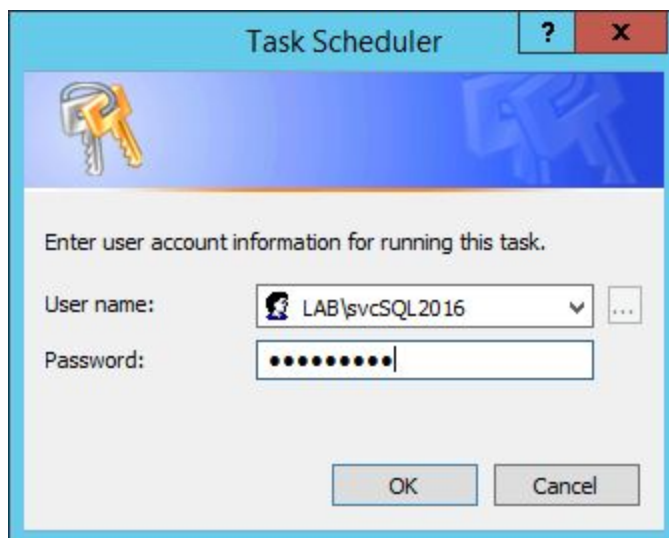
Start in (optional):

OK Cancel

Click **OK** to create the job.

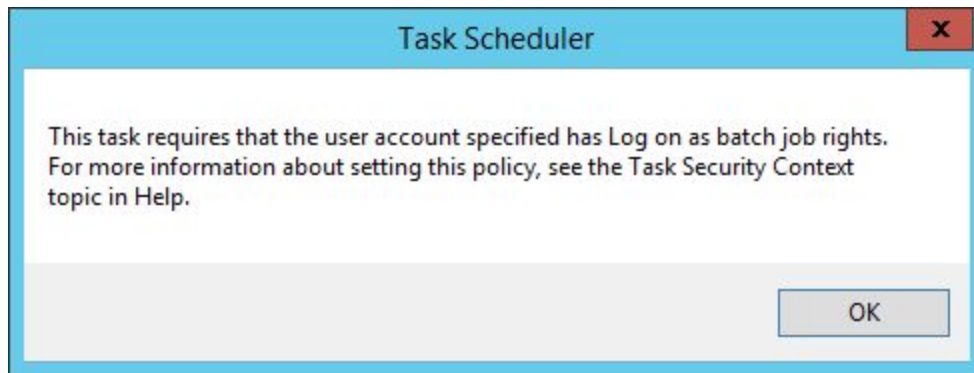


Type in the service account's password and click *OK*.

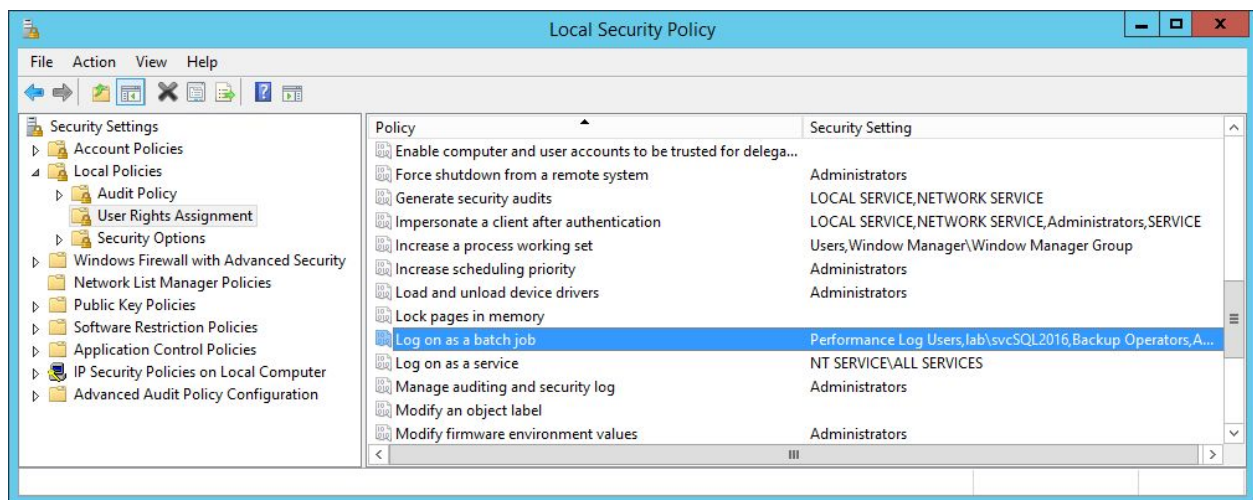


Click *OK* again if you get a message regarding the *Log on as batch job* rights.





If you did get that message, open *Local Security Policy* and navigate to *Local Policies\User Rights Assignment*. Double click on *Log on as a batch job* and add the service account.



Repeat this process for the FULL backup copy job and the DIFF backup copy job if applicable, making sure to change the paths in the command arguments and the schedule.

NOTE: Before creating the Task Scheduler jobs, I removed the job step from the backup jobs on the primary server and redirected the backups to a network share.

An exercise left to the reader is to kick off the Task Scheduler jobs on the backup share server from the primary SQL Server instance, if desired. PowerShell could be helpful here.

## Next Steps Before Going Into Production

See, I told you copying files would be the hardest part.

If you're going to do this in production, you'll also want to consider:

- Monitoring the storage bucket and your local folder to know when they're out of sync
- Monitor network utilization and copy job times - so you know if you're suddenly falling behind, and file copies are taking too long
- Trend backup sizes to know when you'll run into problems with your network bandwidth - before it actually happens

This sounds like a lot of traditional systems administration work - and it is. If you're going to use Google Compute Engine as your disaster recovery site for SQL Server, you're probably going to use it for other applications too. Use one standard set of sysadmin techniques to accomplish the monitoring & trending tasks, and use those across all of your applications.

This is also why we like backing up to a file share, and then letting that file server synchronize the file share up to your storage bucket. You can set up file shares for lots of applications, and then let the sysadmins manage all of your disaster recovery file sync work in one place.

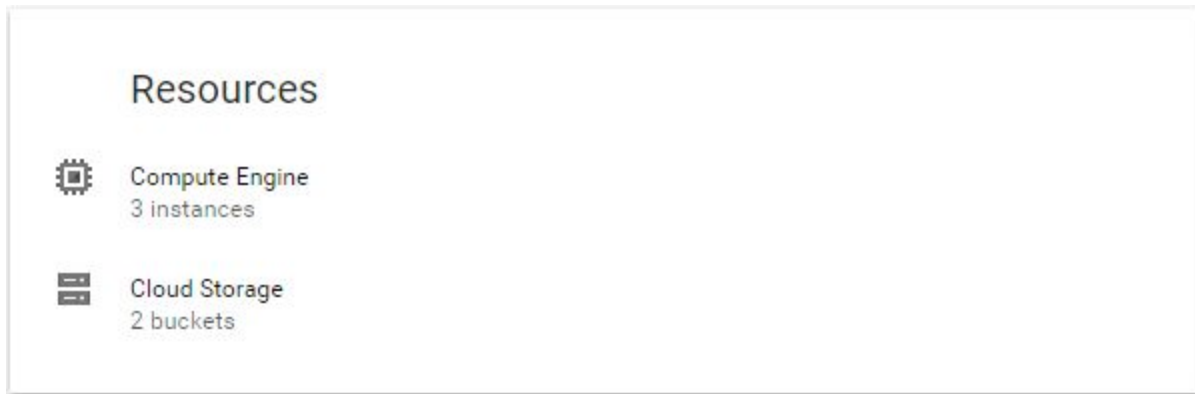
Don't call it a single point of failure, though. Managers hate that. Call it a single point of success.

# Failing Over to GCE

Now let's simulate a disaster where you want to bring up a new server in GCE and restore the database using the files in the storage bucket.

## Creating a VM Using the UI

From the GCP Dashboard, select *Compute Engine* in the *Resources* box.





Click *CREATE INSTANCE*.

 [CREATE INSTANCE](#)

Give it a name, specify which zone it should be in and select the type of machine you want.

For Zone, build your SQL Server in the same zone as your storage bucket in order to maximize file copy throughput and get back online faster.


Name 


Zone 

Machine type  
 15 GB memory [Customize](#)

Change *Boot disk* to a custom image by clicking *Change* and then either *OS images* or *Custom images*, depending on your licensing.

We will be using SQL Server 2016 on Windows 2016 with a 200GB SSD. (Smaller boot drives run the risk of filling up due to Windows Updates, or that idiotic sysadmin you work with who keeps saving ISO files to his desktop, not realizing that it goes to a folder on the C drive.)

Boot disk 



New 10 GB standard persistent disk  
Image  
Debian GNU/Linux 8 (jessie)

Change

---

OS images   Application images   Custom images   Snapshots   Existing disks

- ☐ Debian GNU/Linux 8 (jessie)  
amd64 built on 2016-12-15
- ☐ CentOS 6  
x86\_64 built on 2016-12-12
- ☐ CentOS 7  
x86\_64 built on 2016-12-12
- ☐ CoreOS alpha 1262.0.0  
amd64-usr published on 2016-12-15
- ☐ CoreOS beta 1235.2.0  
amd64-usr published on 2016-12-07
- ☐ CoreOS stable 1185.5.0  
amd64-usr published on 2016-12-07
- ☐ Ubuntu 12.04 LTS  
amd64 precise image built on 2016-12-05
- ☐ Ubuntu 14.04 LTS  
amd64 trusty image built on 2016-12-13
- ☐ Ubuntu 16.04 LTS  
amd64 xenial image built on 2016-12-21
- ☐ Ubuntu 16.10  
amd64 yakkety image built on 2016-12-05
- ☐ Red Hat Enterprise Linux 6  
x86\_64 built on 2016-12-12
- ☐ Red Hat Enterprise Linux 7  
x86\_64 built on 2016-12-12
- ☐ SUSE Linux Enterprise Server 11 SP4  
x86\_64 built on 2016-12-21
- ☐ SUSE Linux Enterprise Server 12 SP2  
x86\_64 built on 2016-12-14
- ☐ Windows Server 2008 R2  
Server with Desktop Experience, x64 built on 2016-12-13
- ☐ Windows Server 2012 R2  
Server with Desktop Experience, x64 built on 2016-12-13
- ☒ Windows Server 2016  
Server with Desktop Experience, x64 built on 2016-12-13

In this same screen, you'll configure the type and size of your SSD:

|                       |             |
|-----------------------|-------------|
| Boot disk type ?      | Size (GB) ? |
| SSD persistent disk ▼ | 200         |

Expand *Management*, *disk*, *networking*, *SSH keys* and then select *Networking*.

⌵ [Management, disk, networking, SSH keys](#)

Specify the *Network* you created and which *Subnetwork* this VM should be in. Use a custom unused *Internal IP address* with no *External IP*. (We're not going to expose this SQL Server to the evil darkness that is the public Internet.) Enable *IP forwarding*.

|                     |       |                   |          |
|---------------------|-------|-------------------|----------|
| Management          | Disks | <b>Networking</b> | SSH Keys |
| Network ?           |       |                   |          |
| wsfcnet ▼           |       |                   |          |
| Subnetwork ?        |       |                   |          |
| wsfcsubnet1 ▼       |       |                   |          |
| Internal IP ?       |       |                   |          |
| Custom ▼            |       |                   |          |
| Internal IP address |       |                   |          |
| 10.0.0.14           |       |                   |          |
| External IP ?       |       |                   |          |
| None ▼              |       |                   |          |
| IP forwarding ?     |       |                   |          |
| On ▼                |       |                   |          |

It's time to *Create* the instance.

You will be billed for this instance. [Learn more](#)

|        |        |
|--------|--------|
| Create | Cancel |
|--------|--------|

## Creating a VM Using the Command Line

Use *Google Cloud Shell* to run the commands. The shell is available in the top right corner of the page, or you can run it locally by installing the [Google Cloud SDK](#). (Yes, it even runs on other operating systems - Brent runs it on his Macs.)



First, create a disk and a Windows image with the *guestOSFeatures* enabled.

```
gcloud compute disks create windows-server-2016-disk --size 200 --zone us-central1-f --type pd-ssd --image /windows-cloud/windows-server-2016-dc-v20161213
```

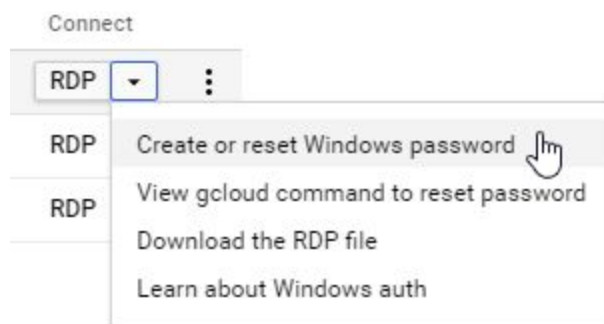
```
gcloud alpha compute images create windows-server-2016 --source-disk windows-server-2016-disk --source-disk-zone us-central1-f --guest-os-features MULTI_IP_SUBNET
```

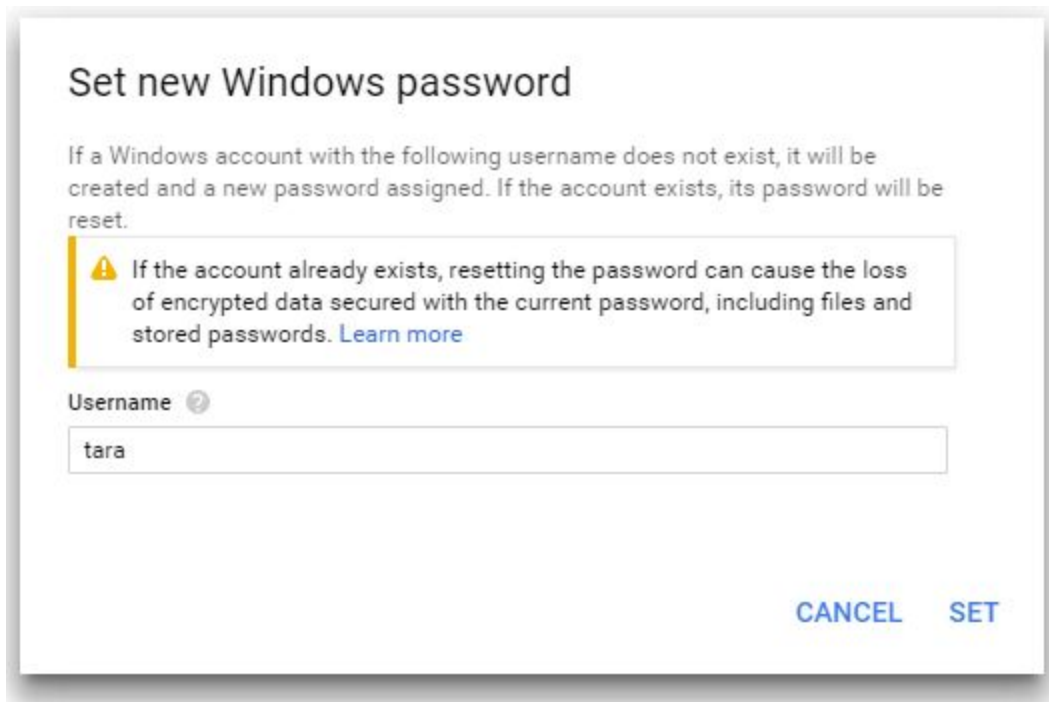
Then create the VM using the image just created.

```
gcloud compute instances create bou-sql1 --machine-type n1-standard-4 --boot-disk-type pd-ssd --boot-disk-size 200GB --image windows-server-2016 --zone us-central1-f --subnet wsfcsbnet1 --private-network-ip=10.0.0.4 --can-ip-forward
```

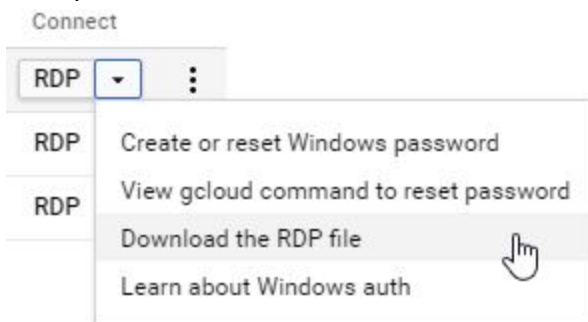
## Connecting to the VM

Create a local user and then RDP to the VM.





To RDP to a VM, you'll need to download the RDP file and then use that to connect. (Mac users can open this file with the free [Microsoft Remote Desktop app](#) from the Mac App Store.)



Note: If you restart the server, you'll need to download a new file since the external IP will have changed.

Double click on the downloaded RDP file and use the local account you created to connect.

## Configuring the VM

RDP to the VM and update all components to the latest version, such as .NET Framework and then install SQL Server.

You may want to switch to a static IP, but the network part is left up to the reader. We are focusing on SQL Server here.

Add logins, jobs and anything else that is not stored inside the user database. You may want to automate creating scripts for those and push them to the storage bucket too. That exercise is left to the reader too.

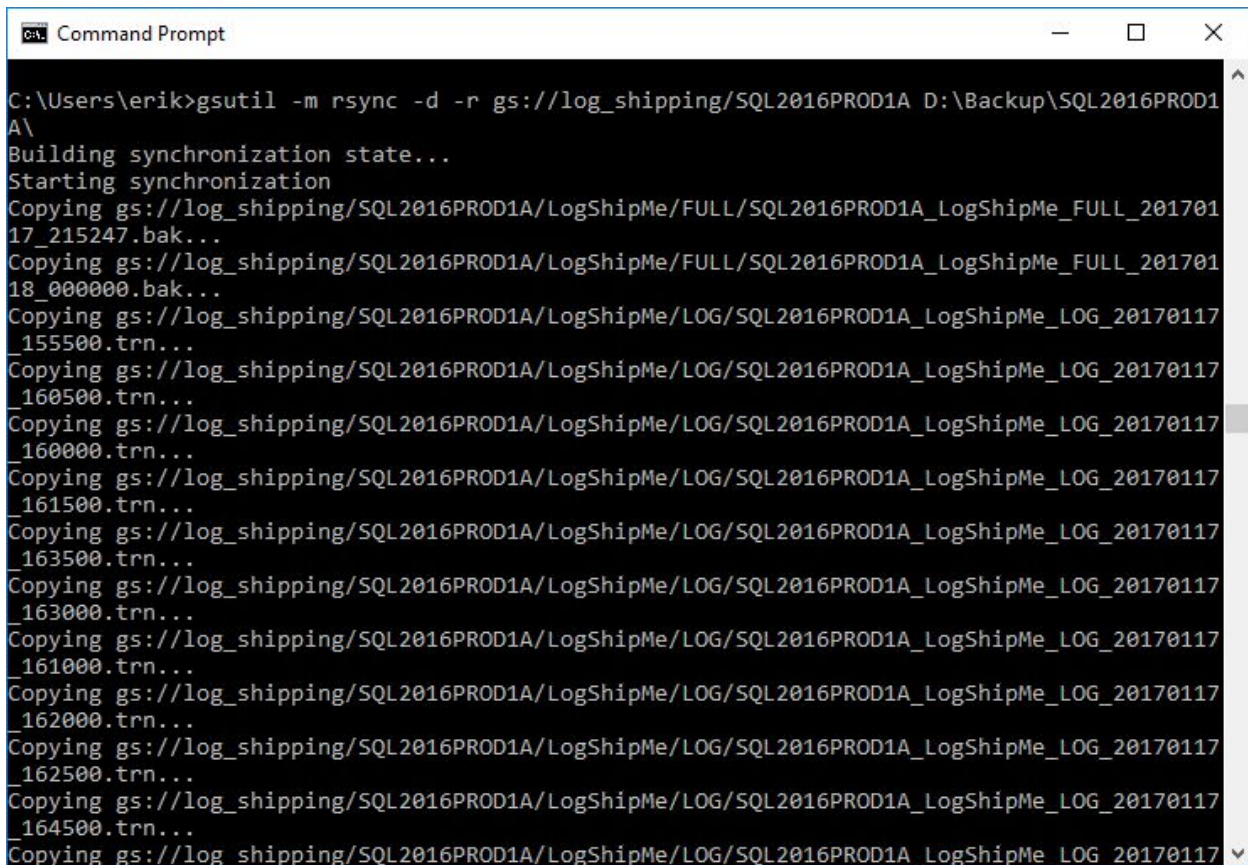
Install Google Cloud SDK as we described earlier in the white paper.

Create a folder for the files that will be downloaded.

D:\Backup\SQL2016PROD1A\

Download the files from the storage bucket. The -m option will allow it to download the files in parallel, which is helpful here since we could be downloading a lot of files.

gsutil -m rsync -d -r gs://log\_shipping/SQL2016PROD1A D:\Backup\SQL2016PROD1A\



```
Command Prompt
C:\Users\erik>gsutil -m rsync -d -r gs://log_shipping/SQL2016PROD1A D:\Backup\SQL2016PROD1A\
Building synchronization state...
Starting synchronization
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/FULL/SQL2016PROD1A_LogShipMe_FULL_20170117_215247.bak...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/FULL/SQL2016PROD1A_LogShipMe_FULL_20170118_000000.bak...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_155500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_160500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_160000.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_161500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_163500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_163000.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_161000.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_162000.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_162500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_164500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170117_165000.trn...
```



```
Copy Command Prompt
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_151001.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_151500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_152500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_153000.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_153500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_154000.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_154500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_155000.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_160000.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_155500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_160500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_161500.trn...
Copying gs://log_shipping/SQL2016PROD1A/LogShipMe/LOG/SQL2016PROD1A_LogShipMe_LOG_20170118_161000.trn...
/ [295/295 files][ 29.0 MiB/ 29.0 MiB] 100% Done    1.3 MiB/s ETA 00:00:00
Operation completed over 295 objects/29.0 MiB.
C:\Users\erik>
```

## Restoring the Database

Create the DatabaseRestore stored procedure, which will be used to restore the database.

[DatabaseRestore](#) was originally written by Greg White who used Greg Robidoux's base code from [an article on MSSQLTips](#). It assumes you are using [Ola Hallengren's DatabaseBackup](#) stored procedure for your backups. It could be rewritten to handle other backup solutions though.

We took Greg White's code and made changes to it:

- Removed code that assumed that the FULL backups are copy-only backups
- Removed @BackupPath parameter and added @BackupPathFull and @BackupPathLog in case the files are stored in different base paths
- Removed @LogToTable parameter as it wasn't used in the code
- Added @RunRecovery and @ContinueLogs in case the user needed to restore more LOG backups
- Changed the data types of the input parameters to match system data types or to use the smallest data type

- Added columns to the table variables that store the output of RESTORE FILELISTONLY and RESTORE HEADERONLY since SQL Server 2016 has more columns in the output
- Added code to read the LSN information from the LOG backup and compare it to the LSN from the newest FULL backup so that it doesn't fail when it tries to restore a LOG backup that is too early
- Added code to read the LSN information from the restored database and compare it to the LSN from the LOG backups when @ContinueLogs = 1 so that it can figure out which LOG file to restore and not throw an error for the LOG backups that were already restored
- Cleaned up the code to have consistency between upper and lower cases of variables

DatabaseRestore may fail if not using SQL Server 2016 due to the extra columns that were added to @Headers and @FileListParameters.

#### Parameters:

@Database NVARCHAR(128) - name of the source database  
 @RestoreDatabaseName NVARCHAR(128), default=NULL - name of the restored database, can leave off or set to NULL if the restored name will be the source database's name  
 @BackupPathFull NVARCHAR(MAX) - full path with ending backslash where the FULL backups are stored  
 @BackupPathLog NVARCHAR(MAX) - full path with ending backslash where the LOG backups are stored  
 @MoveFiles BIT, default=0 - whether or not you want to use a different location for the database files than what was used on the source server, leave off or set to 1 if using the same path as the source  
 @MoveDataDrive NVARCHAR(260), default=NULL - new location for the data file(s), used when @MoveFiles=1  
 @MoveLogDrive NVARCHAR(260), default=NULL - new location for the log file, used when @MoveFiles=1  
 @TestRestore BIT, default=0 - whether or not you are testing restores, if set to 1 then it drops the database at the end  
 @RunCheckDB BIT, default=0 - whether or not you want it to run DBCC CHECKDB after the restore, it assumes you are using Ola Hallengren's [DatabaseIntegrityCheck](#) stored procedure  
 @ContinueLogs, default=0 - whether or not you are continuing to restore logs after the database has already been restored without recovering it  
 @RunRecovery BIT, default=0 - whether or not to recover the database (RESTORE DATABASE WITH RECOVERY so that it is now usable)

If all files have been downloaded, execute DatabaseRestore to restore the database up to the last downloaded LOG backup and then skip down to where it says "The database is now ready for production usage."

```
EXEC dbo.DatabaseRestore
    @Database = 'LogShipMe',
    @BackupPathFull = 'D:\Backup\SQL2016PROD1A\LogShipMe\FULL\',
    @BackupPathLog = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\',
    @ContinueLogs = 0,
    @RunRecovery = 1;
```



If instead the FULL backup and only some of the LOG files have been downloaded and you want to get started on the restore, execute it with @RunRecovery=0.

```
EXEC dbo.DatabaseRestore
```

```

@Database = 'LogShipMe',
@BackupPathFull = 'D:\Backup\SQL2016PROD1A\LogShipMe\FULL\',
@BackupPathLog = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\',
@ContinueLogs = 0,
@RunRecovery = 0;

```

```

EXEC dbo.DatabaseRestore
    @Database = 'LogShipMe',
    @BackupPathFull = 'D:\Backup\SQL2016PROD1A\LogShipMe\FULL\',
    @BackupPathLog = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\',
    @ContinueLogs = 0,
    @RunRecovery = 0;

```

Messages

```

RESTORE DATABASE LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\FULL\SQL2016PROD1A_LogShipMe_FULL_20170119_000001.bak' WITH NORECOVERY, REPLACE

Date and time: 2017-01-19 19:26:04
Command: RESTORE DATABASE LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\FULL\SQL2016PROD1A_LogShipMe_FULL_20170119_000001.bak' WITH NORECOVERY, REPLACE

Processed 336 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 2 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE DATABASE successfully processed 338 pages in 0.218 seconds (12.088 MB/sec).
Outcome: Succeeded
Duration: 00:00:01
Date and time: 2017-01-19 19:26:05

RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShipMe_LOG_20170119_000500.trn' WITH NORECOVERY

Date and time: 2017-01-19 19:26:05
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShipMe_LOG_20170119_000500.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 2 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE LOG successfully processed 2 pages in 0.042 seconds (0.360 MB/sec).
Outcome: Succeeded
Duration: 00:00:00
Date and time: 2017-01-19 19:26:05

RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShipMe_LOG_20170119_001000.trn' WITH NORECOVERY

Date and time: 2017-01-19 19:26:05
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShipMe_LOG_20170119_001000.trn' WITH NORECOVERY

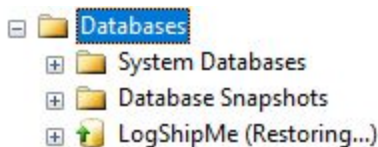
Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 2 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE LOG successfully processed 2 pages in 0.019 seconds (0.462 MB/sec).
Outcome: Succeeded
Duration: 00:00:01
Date and time: 2017-01-19 19:26:06

RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShipMe_LOG_20170119_001500.trn' WITH NORECOVERY

Date and time: 2017-01-19 19:26:06
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A_LogShipMe_LOG_20170119_001500.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 1 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE LOG successfully processed 1 pages in 0.016 seconds (0.122 MB/sec).
Outcome: Succeeded
Duration: 00:00:00
Date and time: 2017-01-19 19:26:06

```



If more files have been downloaded but there are still more to download, execute it again but with @ContinueLogs=1.

```

EXEC dbo.DatabaseRestore
    @Database = 'LogShipMe',
    @BackupPathFull = 'D:\Backup\SQL2016PROD1A\LogShipMe\FULL\',

```



```
@BackupPathLog = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG',
@ContinueLogs = 1,
@RunRecovery = 0;
```

```
EXEC dbo.DatabaseRestore
@Database = 'LogShipMe',
@BackupPathFull = 'D:\Backup\SQL2016PROD1A\LogShipMe\FULL\%',
@BackupPathLog = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\%',
@ContinueLogs = 1,
@RunRecovery = 0;
```

---

100 %

Messages

Duration: 00:00:00  
Date and time: 2017-01-19 19:30:25

RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A\_LogShipMe\_LOG\_20170119\_191000.trn' WITH NORECOVERY

Date and time: 2017-01-19 19:30:25  
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A\_LogShipMe\_LOG\_20170119\_191000.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.  
Processed 1 pages for database 'LogShipMe', file 'LogShipMe\_log' on file 1.  
RESTORE LOG successfully processed 1 pages in 0.014 seconds (0.139 MB/sec).  
Outcome: Succeeded  
Duration: 00:00:01  
Date and time: 2017-01-19 19:30:26

RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A\_LogShipMe\_LOG\_20170119\_191500.trn' WITH NORECOVERY

Date and time: 2017-01-19 19:30:26  
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A\_LogShipMe\_LOG\_20170119\_191500.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.  
Processed 1 pages for database 'LogShipMe', file 'LogShipMe\_log' on file 1.  
RESTORE LOG successfully processed 1 pages in 0.014 seconds (0.139 MB/sec).  
Outcome: Succeeded  
Duration: 00:00:00  
Date and time: 2017-01-19 19:30:26

RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A\_LogShipMe\_LOG\_20170119\_192000.trn' WITH NORECOVERY

Date and time: 2017-01-19 19:30:26  
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A\_LogShipMe\_LOG\_20170119\_192000.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.  
Processed 1 pages for database 'LogShipMe', file 'LogShipMe\_log' on file 1.  
RESTORE LOG successfully processed 1 pages in 0.020 seconds (0.097 MB/sec).  
Outcome: Succeeded  
Duration: 00:00:00  
Date and time: 2017-01-19 19:30:26

RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A\_LogShipMe\_LOG\_20170119\_192500.trn' WITH NORECOVERY

Date and time: 2017-01-19 19:30:26  
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG\SQL2016PROD1A\_LogShipMe\_LOG\_20170119\_192500.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.  
Processed 1 pages for database 'LogShipMe', file 'LogShipMe\_log' on file 1.  
RESTORE LOG successfully processed 1 pages in 0.015 seconds (0.130 MB/sec).  
Outcome: Succeeded  
Duration: 00:00:00  
Date and time: 2017-01-19 19:30:26

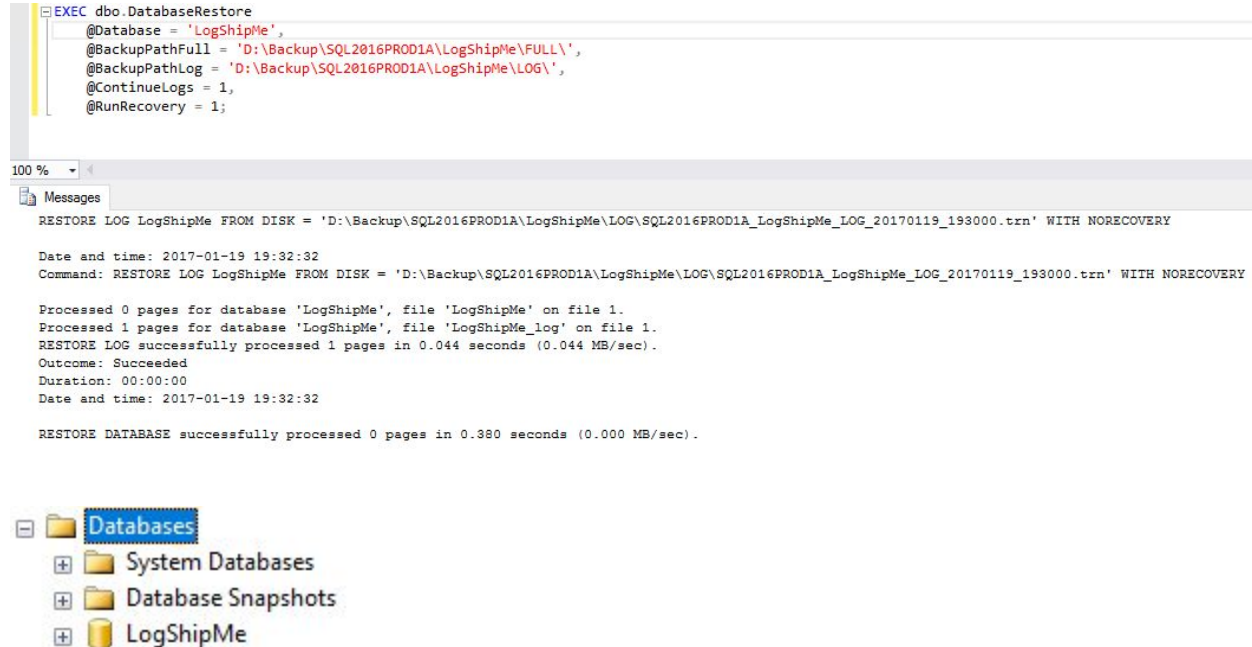
Databases

- System Databases
- Database Snapshots
- LogShipMe (Restoring...)

When the last of the files have been downloaded, execute it a final time with @ContinueLogs=1 and @RunRecovery=1.

```
EXEC dbo.DatabaseRestore
@Database = 'LogShipMe',
```

```
@BackupPathFull = 'D:\Backup\SQL2016PROD1A\LogShipMe\FULL',  
@BackupPathLog = 'D:\Backup\SQL2016PROD1A\LogShipMe\LOG',  
@ContinueLogs = 1,  
@RunRecovery = 1;
```



The database is now ready for production usage.

Setup jobs to copy the backup files into the storage bucket.

```
"C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\gsutil.cmd" rsync -d -r  
D:\Backup\BOU-SQL1\LogShipMe\FULL\ gs://log_shipping/BOU-SQL1/LogShipMe/FULL/
```

```
"C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\gsutil.cmd" rsync -d -r  
D:\Backup\BOU-SQL1\LogShipMe\LOG\ gs://log_shipping/BOU-SQL1/LogShipMe/LOG/
```

# Failing Back to the Primary Server

When you are ready to fail back to the primary server, you can failback with minimal downtime. Who doesn't like that? You can be the knight in shining armor.

The backups for the VM are already being copied to the storage bucket if you setup the two jobs at the end of the last section, so you are ready to start downloading the backups and restoring the database on the primary server.

Create the folders on the primary server where the backups will be downloaded to.

D:\MSSQL\Backup\BOU-SQL1\LogShipMe\



Download the FULL backup and all of the LOG backups thus far.

```
"C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\gsutil.cmd" -m rsync -d -r  
gs://log_shipping/BOU-SQL1\LogShipMe/ D:\MSSQL\Backup\BOU-SQL1\LogShipMe\
```

```
Command Prompt
C:\Users\tara>"C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\gsutil.cmd" -m rsync -d -r gs://log_shipping/BOU-SQL1/LogShipMe/ D:\MSSQL\Backup\BOU-SQL1\LogShipMe\
Building synchronization state...
Starting synchronization
Copying gs://log_shipping/BOU-SQL1/LogShipMe/FULL/BOU-SQL1_LogShipMe_FULL_20170119_194559.bak...
- [0/35 files] 0.0 B/ 8.3 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_172500.trn...
- [0/35 files] 0.0 B/ 8.3 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_172000.trn...
- [0/35 files] 0.0 B/ 8.3 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/FULL/BOU-SQL1_LogShipMe_FULL_20170119_171532.bak...
- [0/35 files] 0.0 B/ 8.3 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_173000.trn...
- [0/35 files] 0.0 B/ 8.3 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_175000.trn...
- [0/35 files] 0.0 B/ 8.3 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_173500.trn...
- [0/35 files] 0.0 B/ 8.3 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_174500.trn...
- [0/35 files] 0.0 B/ 8.3 MiB 0% Done
```

```
Command Prompt
195500.trn...
: [17/35 files] 5.8 MiB/ 8.3 MiB 69% Done
: [18/35 files] 5.8 MiB/ 8.3 MiB 69% Done
: [18/35 files] 6.0 MiB/ 8.3 MiB 72% Done
: [19/35 files] 6.1 MiB/ 8.3 MiB 73% Done
: [20/35 files] 6.5 MiB/ 8.3 MiB 77% Done
: [21/35 files] 7.0 MiB/ 8.3 MiB 83% Done
: [22/35 files] 7.0 MiB/ 8.3 MiB 83% Done
: [23/35 files] 7.1 MiB/ 8.3 MiB 84% Done
: [24/35 files] 7.2 MiB/ 8.3 MiB 86% Done
: [25/35 files] 7.3 MiB/ 8.3 MiB 87% Done
: [26/35 files] 7.4 MiB/ 8.3 MiB 88% Done
: [27/35 files] 7.5 MiB/ 8.3 MiB 89% Done
: [28/35 files] 7.7 MiB/ 8.3 MiB 92% Done
: [29/35 files] 7.7 MiB/ 8.3 MiB 92% Done
: [30/35 files] 7.8 MiB/ 8.3 MiB 93% Done
: [31/35 files] 7.9 MiB/ 8.3 MiB 94% Done
: [32/35 files] 8.0 MiB/ 8.3 MiB 96% Done
: [33/35 files] 8.1 MiB/ 8.3 MiB 97% Done
: [34/35 files] 8.2 MiB/ 8.3 MiB 98% Done
: [34/35 files] 8.2 MiB/ 8.3 MiB 98% Done
/ [34/35 files] 8.3 MiB/ 8.3 MiB 99% Done 54.9 KiB/s ETA 00:00:00
- [35/35 files] 8.3 MiB/ 8.3 MiB 100% Done 54.9 KiB/s ETA 00:00:00

Operation completed over 35 objects/8.3 MiB.
C:\Users\tara>
```

Create the DatabaseRestore stored procedure on the primary server and then run it to start restoring the database without recovering it.

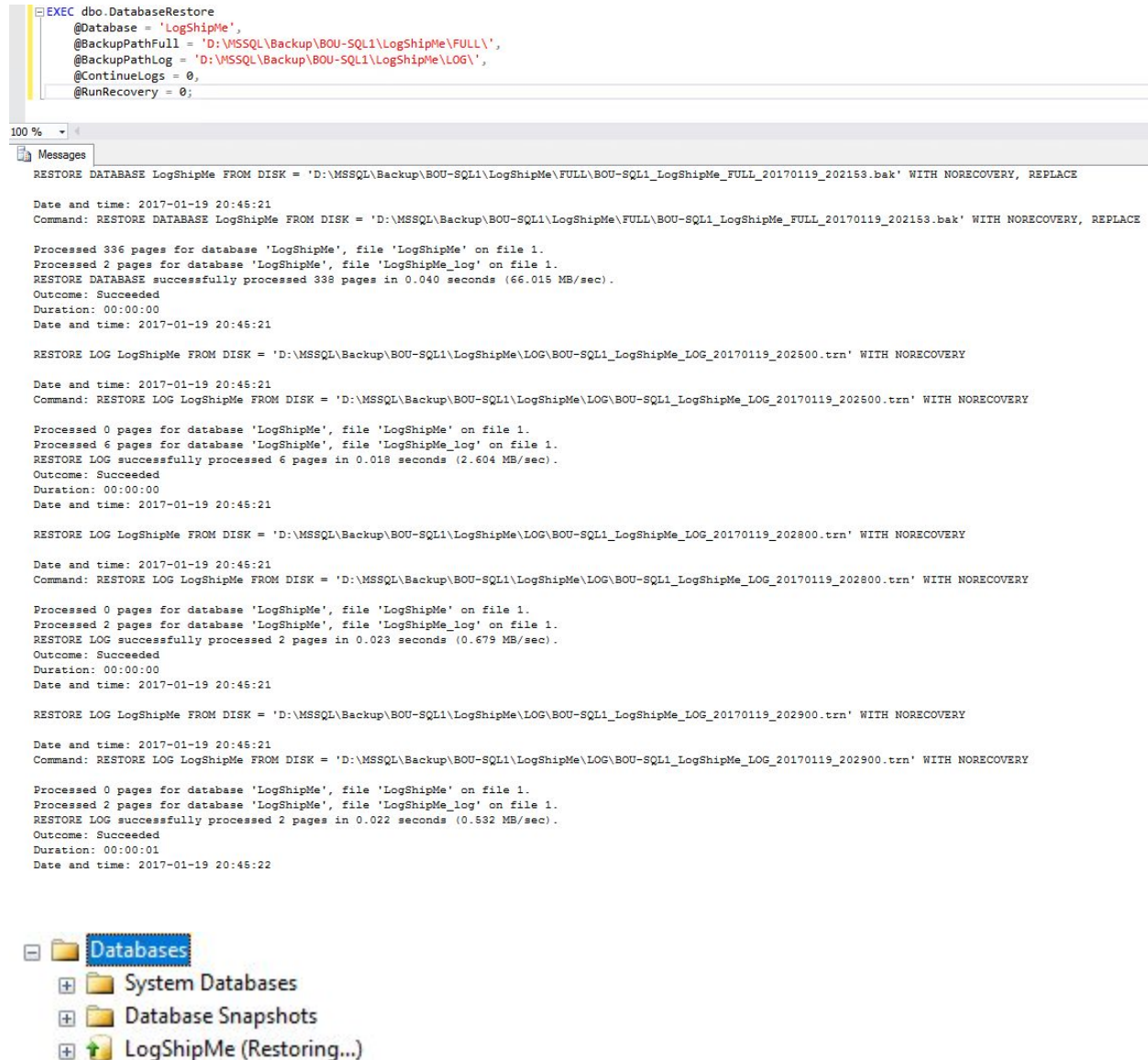
```
EXEC dbo.DatabaseRestore
    @Database = 'LogShipMe',
```



```

@BackupPathFull = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\FULL',
@BackupPathLog = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG',
@ContinueLogs = 0,
@RunRecovery = 0;

```



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a T-SQL query for restoring a database. The bottom pane, titled 'Messages', shows the execution results of the restore command.

```

EXEC dbo.DatabaseRestore
    @Database = 'LogShipMe',
    @BackupPathFull = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\FULL',
    @BackupPathLog = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG',
    @ContinueLogs = 0,
    @RunRecovery = 0;

```

**Messages**

```

RESTORE DATABASE LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\FULL\BOU-SQL1_LogShipMe_FULL_20170119_202153.bak' WITH NORECOVERY, REPLACE

Date and time: 2017-01-19 20:45:21
Command: RESTORE DATABASE LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\FULL\BOU-SQL1_LogShipMe_FULL_20170119_202153.bak' WITH NORECOVERY, REPLACE

Processed 336 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 2 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE DATABASE successfully processed 338 pages in 0.040 seconds (66.015 MB/sec).
Outcome: Succeeded
Duration: 00:00:00
Date and time: 2017-01-19 20:45:21

RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_202500.trn' WITH NORECOVERY

Date and time: 2017-01-19 20:45:21
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_202500.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 6 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE LOG successfully processed 6 pages in 0.018 seconds (2.604 MB/sec).
Outcome: Succeeded
Duration: 00:00:00
Date and time: 2017-01-19 20:45:21

RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_202800.trn' WITH NORECOVERY

Date and time: 2017-01-19 20:45:21
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_202800.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 2 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE LOG successfully processed 2 pages in 0.023 seconds (0.679 MB/sec).
Outcome: Succeeded
Duration: 00:00:00
Date and time: 2017-01-19 20:45:21

RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_202900.trn' WITH NORECOVERY

Date and time: 2017-01-19 20:45:21
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_202900.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 2 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE LOG successfully processed 2 pages in 0.022 seconds (0.532 MB/sec).
Outcome: Succeeded
Duration: 00:00:01
Date and time: 2017-01-19 20:45:22

```

**Databases**

- System Databases
- Database Snapshots
- LogShipMe (Restoring...)

Leading up to the maintenance window, continue downloading the backups and running DatabaseRestore to catch up on any LOG backups that have not yet been restored on the primary server.

```

"C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\gsutil.cmd" -m rsync -d -r
gs://log_shipping/BOU-SQL1\LogShipMe\LOG/
D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\

```

```
Command Prompt
C:\Users\tara>"C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\gsutil.cmd" -m rsync -d -r gs://log_shipping/BOU-SQL1/LogShipMe/LOG/ D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\
Building synchronization state...
Starting synchronization
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_203000.trn...
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_203301.trn...
/ [0/16 files] 0.0 B/ 1.7 MiB 0% Done
/ [0/16 files] 0.0 B/ 1.7 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_203100.trn...
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_203401.trn...
/ [0/16 files] 0.0 B/ 1.7 MiB 0% Done
/ [0/16 files] 0.0 B/ 1.7 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_203600.trn...
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_203200.trn...
/ [0/16 files] 0.0 B/ 1.7 MiB 0% Done
/ [0/16 files] 0.0 B/ 1.7 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_203700.trn...
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_203800.trn...
/ [0/16 files] 0.0 B/ 1.7 MiB 0% Done
```

```
Command Prompt
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_204100.trn...
- [0/16 files] 0.0 B/ 1.7 MiB 0% Done
- [0/16 files] 0.0 B/ 1.7 MiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_204300.trn...
- [0/16 files] 0.0 B/ 1.7 MiB 0% Done
- [1/16 files] 108.0 KiB/ 1.7 MiB 6% Done
- [2/16 files] 216.0 KiB/ 1.7 MiB 12% Done
- [3/16 files] 324.0 KiB/ 1.7 MiB 18% Done
- [4/16 files] 432.0 KiB/ 1.7 MiB 25% Done
- [5/16 files] 540.0 KiB/ 1.7 MiB 31% Done
- [6/16 files] 648.0 KiB/ 1.7 MiB 37% Done
- [7/16 files] 756.0 KiB/ 1.7 MiB 43% Done
- [8/16 files] 864.0 KiB/ 1.7 MiB 50% Done
- [9/16 files] 972.0 KiB/ 1.7 MiB 56% Done
- [10/16 files] 1.1 MiB/ 1.7 MiB 62% Done
- [11/16 files] 1.2 MiB/ 1.7 MiB 68% Done
- [12/16 files] 1.3 MiB/ 1.7 MiB 75% Done
- [13/16 files] 1.4 MiB/ 1.7 MiB 81% Done
- [14/16 files] 1.5 MiB/ 1.7 MiB 87% Done
- [15/16 files] 1.6 MiB/ 1.7 MiB 93% Done
- [16/16 files] 1.7 MiB/ 1.7 MiB 100% Done

Operation completed over 16 objects/1.7 MiB.

C:\Users\tara>
```

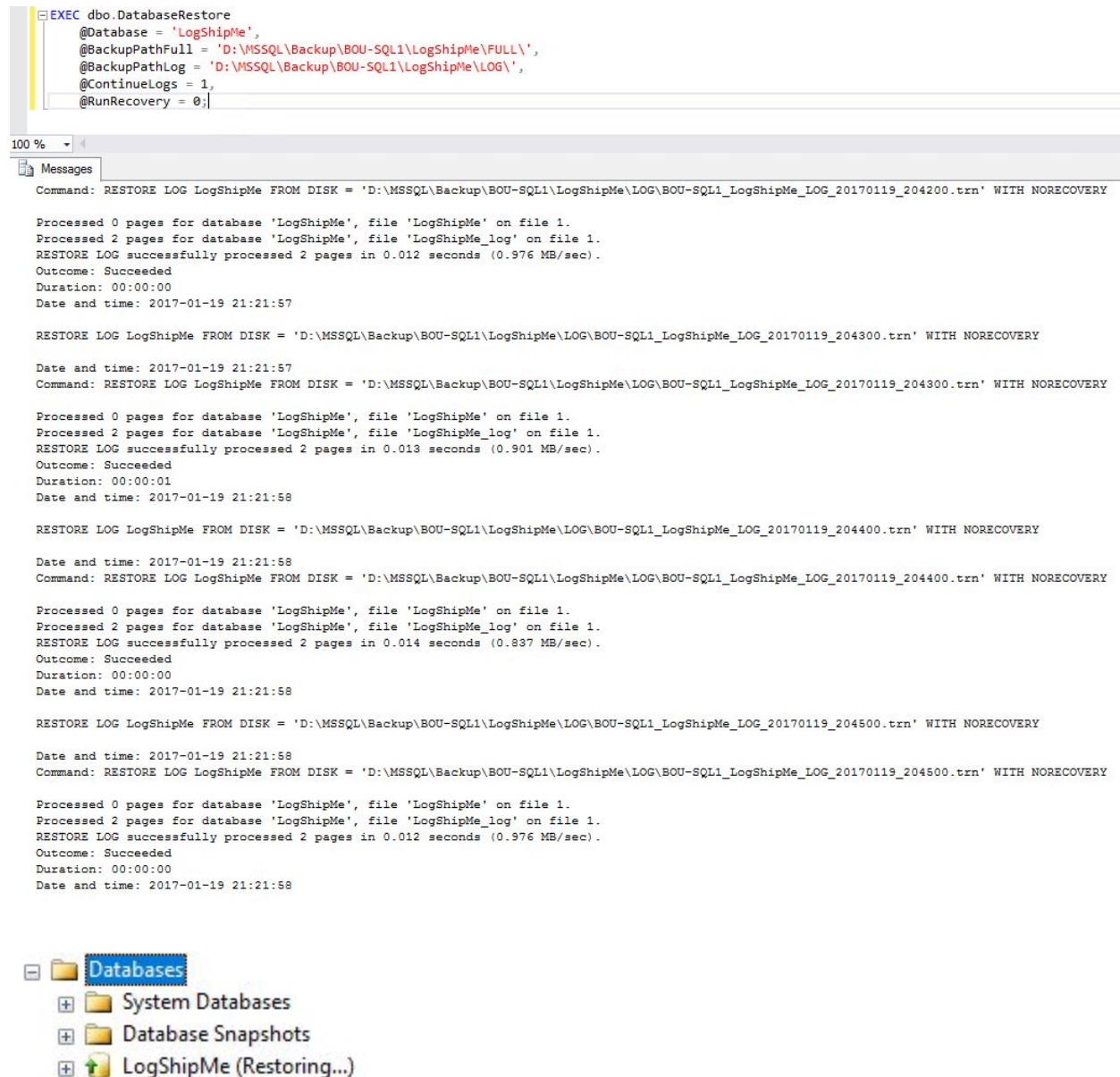
EXEC dbo.DatabaseRestore

@Database = 'LogShipMe',

@BackupPathFull = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\FULL',

@BackupPathLog = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG',

```
@ContinueLogs = 1,  
@RunRecovery = 0;
```



Once the maintenance window has started, on the GCE VM run the LOG backup job and then the copy to the cloud job.

Download the latest backups on the primary server and then restore them with recovery.

```
"C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\gsutil.cmd" -m rsync -d -r  
gs://log_shipping/BOU-SQL1\LogShipMe\LOG/  
D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\
```



```
Command Prompt
C:\Users\tara>"C:\Program Files (x86)\Google\Cloud SDK\google-cloud-sdk\bin\gsutil.cmd" -m rsync -d -r gs://log_shipping/BOU-SQL1/LogShipMe/LOG/ D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\
Building synchronization state...
Starting synchronization
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_213400.trn...
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_213300.trn...
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_213501.trn...
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_213900.trn...
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_213200.trn...
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_213800.trn...
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_213700.trn...
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_213601.trn...
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
Copying gs://log_shipping/BOU-SQL1/LogShipMe/LOG/BOU-SQL1_LogShipMe_LOG_20170119_214000.trn...
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
/ [0/9 files] 0.0 B/972.0 KiB 0% Done
/ [1/9 files] 108.0 KiB/972.0 KiB 11% Done
/ [2/9 files] 216.0 KiB/972.0 KiB 22% Done
- [3/9 files] 324.0 KiB/972.0 KiB 33% Done
- [4/9 files] 432.0 KiB/972.0 KiB 44% Done
- [5/9 files] 540.0 KiB/972.0 KiB 55% Done
- [6/9 files] 648.0 KiB/972.0 KiB 66% Done
- [7/9 files] 756.0 KiB/972.0 KiB 77% Done
- [8/9 files] 864.0 KiB/972.0 KiB 88% Done
- [9/9 files] 972.0 KiB/972.0 KiB 100% Done

Operation completed over 9 objects/972.0 KiB.
```

EXEC dbo.DatabaseRestore

```
@Database = 'LogShipMe',
@BackupPathFull = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\FULL',
@BackupPathLog = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG',
@ContinueLogs = 1,
@RunRecovery = 1;
```

```

EXEC dbo.DatabaseRestore
@Database = 'LogShipMe',
@BackupPathFull = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\FULL\',
@BackupPathLog = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\',
@ContinueLogs = 1,
@RunRecovery = 1;

```

100 %

Messages

```

RESTORE LOG successfully processed 2 pages in 0.016 seconds (0.732 MB/sec).
Outcome: Succeeded
Duration: 00:00:00
Date and time: 2017-01-19 21:46:19

RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_213800.trn' WITH NORECOVERY

Date and time: 2017-01-19 21:46:19
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_213800.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 2 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE LOG successfully processed 2 pages in 0.016 seconds (0.732 MB/sec).
Outcome: Succeeded
Duration: 00:00:00
Date and time: 2017-01-19 21:46:19

RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_213900.trn' WITH NORECOVERY

Date and time: 2017-01-19 21:46:19
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_213900.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 2 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE LOG successfully processed 2 pages in 0.012 seconds (0.976 MB/sec).
Outcome: Succeeded
Duration: 00:00:00
Date and time: 2017-01-19 21:46:19

RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_214000.trn' WITH NORECOVERY

Date and time: 2017-01-19 21:46:19
Command: RESTORE LOG LogShipMe FROM DISK = 'D:\MSSQL\Backup\BOU-SQL1\LogShipMe\LOG\BOU-SQL1_LogShipMe_LOG_20170119_214000.trn' WITH NORECOVERY

Processed 0 pages for database 'LogShipMe', file 'LogShipMe' on file 1.
Processed 2 pages for database 'LogShipMe', file 'LogShipMe_log' on file 1.
RESTORE LOG successfully processed 2 pages in 0.012 seconds (0.976 MB/sec).
Outcome: Succeeded
Duration: 00:00:00
Date and time: 2017-01-19 21:46:19

RESTORE DATABASE successfully processed 0 pages in 0.222 seconds (0.000 MB/sec).

```

Databases

- System Databases
- Database Snapshots
- LogShipMe

The database is now ready for production usage on the primary server.

Once your testing is done, make sure you are still copying backups from the primary server into the storage bucket and then delete the GCE VM, which can easily be done in the GCE UI.

|                          |                                     |            |               |                 |            |                |     |   |
|--------------------------|-------------------------------------|------------|---------------|-----------------|------------|----------------|-----|---|
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | bou-sql1   | us-central1-f | 4 vCPUs, 15 GB  | 10.0.0.4   | 104.154.56.152 | RDP | ⋮ |
| <input type="checkbox"/> | <input type="radio"/>               | bou-sql2   | us-central1-f | 4 vCPUs, 15 GB  | 10.1.0.4   | None           | RDP | ⋮ |
| <input type="checkbox"/> | <input type="radio"/>               | bou-sql3   | us-central1-f | 4 vCPUs, 15 GB  | 10.0.0.5   | None           | RDP | ⋮ |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | dc-windows | us-central1-f | 1 vCPU, 3.75 GB | 10.2.0.100 | 130.211.194.65 | RDP | ⋮ |

Start  
Stop  
Reset  
Delete  
New instance group

# Recap and What to Consider Next

In this white paper, we taught you:

- How to set up a storage bucket in Google Cloud
- How to sync your backups up to that storage bucket
- When disaster strikes, how to build a SQL Server and restore your backups using a cool free open source script
- How to migrate back down on-premises after the disaster goes away

We call this Log Shipping 2.0 because like old-school transaction log shipping, it gets a good copy of your backups in another location. It's more cost effective, though, because you don't have to have a SQL Server up and running all the time in the cloud, restoring your backups.

If you want to be able to react to disaster faster and have less downtime, consider:

**Using Log Shipping 1.0.** Build the SQL Server up in GCE, and then leave it running full time, restoring your log backups as they show up in your storage bucket. This is dramatically more expensive, especially if your licensing doesn't give you the ability to run a free SQL Server instance up in GCE. It does reduce your downtime during a failover.

**Scripting the stand-up process.** Use tools like PowerShell to automate standing up an entire SQL Server with your exact configuration requirements and kicking off the restore process. Depending on the size of your backups, you might be able to get this down to a matter of minutes. If you were really ambitious, you could integrate this with a cloud-based monitoring system to watch your on-premises environment, and if it ran into problems, automatically trigger your build-the-DR-site scripts.

**Using a different HA/DR technology.** SQL Server's asynchronous database mirroring and Always On Availability Groups are options up in the cloud, too. Your primary SQL Server could be continuously streaming your delete/update/insert operations up to a hot standby up in the cloud. Of course, the more ambitious you get with this, the more expensive it gets - both in terms of 24/7 running cloud VMs, and also the sysadmin hours required to maintain the solution.

The first step in designing the right HA/DR solution is to ask your business to sit down with you and define [your recovery point objective \(RPO\) and recovery time objective \(RTO\)](#). We've got a helpful worksheet for that in [our First Responder Kit](#), too.

If you'd like our help, we'd love to lend a hand:

- Our specialty, a 3-day SQL Critical Care® - <https://www.brentozar.com/sql-critical-care/>
- Email us at [Help@BrentOzar.com](mailto:Help@BrentOzar.com), or on the web: <https://www.brentozar.com/contact/>