Horizon Reach Documentation

(Formerly Project Heimdall)

Product Version 1.1.x

Flings
OFFICE OF THE CTO

Author: Andrew Morgan / @andyjmorgan / mandrew@vmware.com

Table of Contents

About:

Horizon Reach or Reach as it will be called in the remainder of the document, is a web based, monitoring and alerting fling for VMware Horizon. Horizon Reach is designed to tackle the disconnect in Enterprise environments wherein each Pod in a Cloud Pod Architecture is its own technology domain and fault domain, or a customer is running multiple, disconnected pods, outside of a Cloud Pod Architecture, but would still like to treat them all as one unit of compute.

Often when troubleshooting these fault domains, it can feel like a game of "Whack a mole" jumping from Pod to Pod trying to find a pertinent session, alarm or event to the problem your user is describing. Reach tackles this issue by performing health checking and gathering pertinent errors from each separate environment and displaying them all in a single place, creating an easy location for administrators to monitor the environment, along with providing a detailed first step in the troubleshooting process.

Product Design Goals:

Reach is designed to be your "first port of call" when you have issues, the workflow is simple:

- Open reach
- drill down to the unit of compute or infrastructure component
- jump to the right console.

As reach is already gathering all this data and 99% of customers struggle to use our API's, I've also included a simple REST API, Documented in Swagger to allow you to leverage the data in Reach to do your own reporting, without having to recreate the logic or connect to each pod.

In addition to the above, reach will also give some unique reports that aren't available in product today, such as:

- Unique, Concurrent User high water marks
- Cross pod Licensing counts
- Historical Usage Reporting across:
    - Sites
    - Pods
    - Global Entitlements
    - Desktop Pools
    - Farms

In future versions of reach, expect to see this list grow further with analytics and insights.

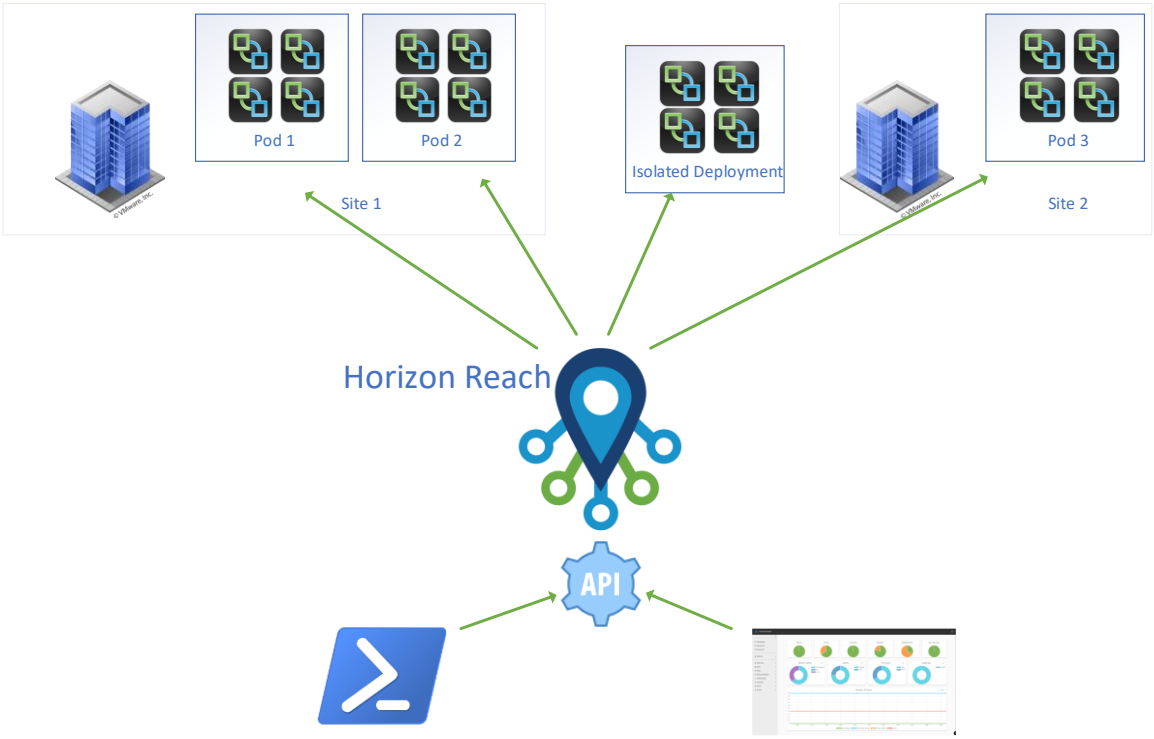
What it's not:

Reach is NOT designed to boil the ocean or recreate existing toolsets today.

Reach will not:

- Gather historical performance data, use VROPS or another monitoring product.
- Replace the Horizon or vSphere console, I don't plan to add driven actions in the near future.

How it works:



For the nerdy or curious, Reach is an ASP .Net Core 2.2 Web API and Angular frontend service, wrapped in an executable (HorizonReach.exe). Horizon Reach can run as a console application (horizonreach.exe –console) or as a service (see installation guide). Horizon Reach also requires the following files:

- Serviceconfiguration.json (no need to guess what that does).
- Local.db (an sqlite database file for historical data).

Once Reach starts, it will:

Check the database exists, is on the right version. It will upgrade / create if needed.
Reads the service configuration file for connections and settings.

At the specified Interval, Reach will "reach" out to each connection server configured (or fail to a backup connection server it learned during the last discovery).

Once connected to a connection server, it will determine if the connection server is a member of a Cloud Pod Architecture. If it is a member of a cloud pod architecture, it will simultaneously connect to all other pods, too.

Once connected it will poll sessions, farms, infrastructure etc and generate a live snapshot to perform health checks against.

Once it has it's snapshots for all infrastructure, it begins a process where it health checks the components (generating alarms) and converts it to the model view for the application. I do this for two reasons:

The horizon classes are convoluted at times, I make them simpler to display in the web app.

By doing this directly after a poll, it allows me to ensure the web app is always lightning fast, ensuring as little computation is done during page load times.

Only after this process has completed, the new data is displayed in the web UI. Parallel to this, the database update procedure is completed if the time between the last database update has lapsed.

Both of these objects are visible / downloadable via settings > web service settings > download snapshot.

It's worth noting that Horizon Reach is completely stateless, if a pod cannot be reached it will disappear from the console at the next poll, replaced with a connectivity error.

Product Requirements:

In order to successfully and easily deploy Horizon reach, you will need:

Supported servers:

- Windows Server 2016
- Windows Server 2019
- Windows 10 1809 (x64) (don't use this for large scale)

VMware Horizon (all license editions):

- VMware Horizon 7.5.1 (bare minimum)
- VMware Horizon 7.6 (better)
- VMware Horizon 7.10 (best)

Browser:

Tested on Chromium (Google Chrome and Brave) and safari. Will probably work with Firefox too, but I haven't tested. I have no intention of testing or supporting Internet Explorer at all or Edge until it ports to Chromium.

Tested Scale:

These sizing are best estimate currently, during the fling Alpha and Beta phase, up to 20 pods and 30,000 user sessions were scale tested and performed well. That being said, it is imperative that you monitor the performance of the server as you scale up and deploy.

The key metric is bound to be CPU and you will see big spikes when Heimdall performs it's timed collection, data crunching and updating.

This frequency (settings > web service settings > Query Interval) will need to be compared to the last collection job. If you take that the last collection job took 60 seconds, multiply this number by 2.5 and set your polling interval to this.
E.G: if the web service discovery takes 40 seconds, then the recommended poll time would be 100 seconds.

This will ensure the best performance of Reach and also reduce the load on the connection servers during polling.

You'll also notice a larger spike when the database collection is running (settings > web service settings > Performance Interval). This is to be expected and I highly recommend you don't perform this more frequently than 15-minute intervals.

Sizing:

As with the above data, sizing for Reach is dependent on the volume of objects within each pod. Please take these figures as a loose guess and monitor performance to ensure the server has the resources it requires.

Size for 1 - 5 pods:

- 4 CPU
- 4 GB Ram

Size for 5 - 10 pods:

- 4 CPU
- 6 GB Ram

Size for 10 + pods *:

- 6+ CPU

- 10gb+ Ram

**Note: at this scale, you'd be better off reaching out to me via email and I'll assist with the scaling. (contact details above, I won't mind at all so don't be shy).*

Installation:

Run the latest setup.exe downloaded from the flings site on the server you wish to gather data.

Uninstalling Horizon Reach:

Remove the application from "Add Remove Programs" (appwiz.cpl).

Backing up Horizon Reach:

To backup Horizon Reach, backup:

- Serviceconfiguration.json.
- Local.db.
- Any custom certificate you may have added to the local machine store.

Updating Horizon Reach:

With the release of 1.1, a new installer is available instead of the old zip file and powershell script installation process. If you wish to upgrade from a previous version, please follow the below steps:

Upgrading from pre 1.1.2x:

- Stop the existing Reach service
- Run the "Remove-reachservice.ps1" powershell script from the current directory
- Backup the directory (zip it).
- run the new installer.
- stop the new service.
- copy serviceconfiguration.json and local.db from the old installation folder to the new.
- Start the new service.
- Logon and make sure it's all working.
- If all ok, delete the old location.

Rollback:

- Uninstall the latest version
- Run "Install-reachservice.ps1" from the old installation directory

Default Credentials:

Username: administrator
Password: Heimdall123

Username: viewer
Password: Heimdall123 (don't use this for setup).

Configuring Horizon Reach for the first time:

Once deployed, browse to the server's name or IP Address on port 9443, e.g.:

Https://localhost:9443

If all has gone to plan, you'll hit an ssl error first (self-signed certificate) then a logon page:

Once logged in, you will receive a wizard to setup Horizon reach for the first time. The wizard is self-explanatory, but just in case:

Step 1: The URL should be https://YourConnectionServerName.domain.local
Note: if you add a CPA connection, you don't need to add other pods, Horizon Reach will figure it out from there.

Step 2: Set the name to something you will remember.

Step 3: Add a description, whomever takes your job when you leave will thank you.

Step 4 Use a read only administrator account, don't use your credentials dude, Horizon Reach won't be sorry when it locks your account after a password change.

Step 5: test the connection, if all goes to plan, save, if not, check what you got wrong.

Note: if your SSL certificate is untrusted, you'll be notified of this and the thumbprint will be automatically added to the connection, if you're happy the certificate is right, simply try again with the thumbprint and it should work great.

Once configured, Horizon Reach will give you a swirly whirly progress screen while it attempts to gather data, eventually giving you your new, shiny dashboard.

But what do I do if it fails?

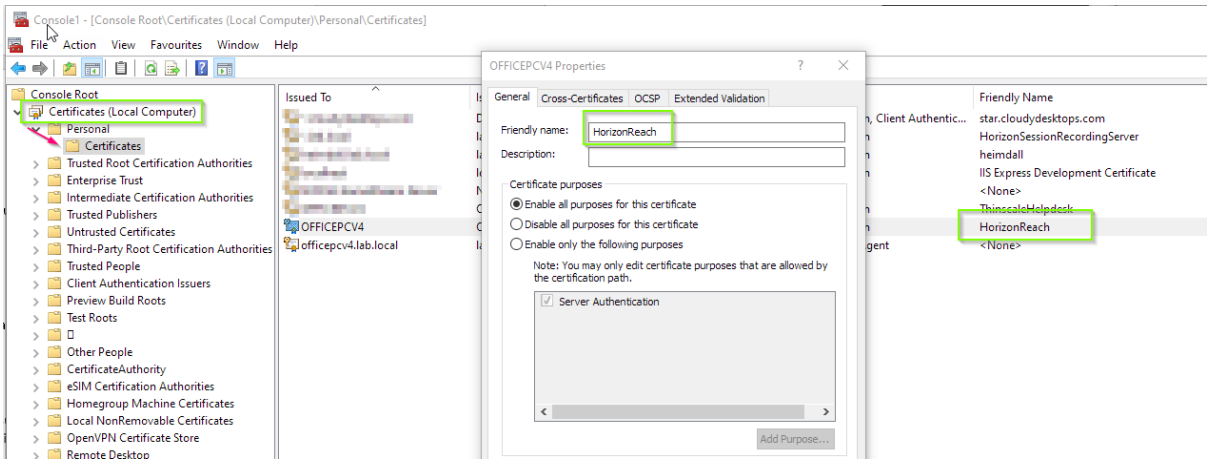
Well aside from checking the requirements, url, credentials, etc. Head into the installation directory > logs and check the HorizonDiscovery.log file. Reach out to me if you're struggling via email or flings site.

Things you can't do in the UI:

The web app is self-explanatory, but there are some features that just didn't make the cut for 1.0 web app. The following details are below:

Modifying SSL Certificate:

If you want to change the certificate to a trusted certificate (which you really should, lazy pants) you can modify the certificate, in the local machines store. Assign the friendly name "HorizonReach" to the certificate you wish to use and restart the Horizon Reach service.



Adding LDAPS support:

If you wish to enable LDAPS support (ldap on 389 is not supported) you can enable this in the service configuration file. You must authentication with the UserPrincipalName e.g.:

Username@domain.local

If you don't, it will attempt to authenticate you against the built-in accounts, which won't work.

IMPORTANT: DON'T USE THE DOMAIN USERS GROUP, IT WONT WORK.

Requirements:

- A Domain Controller infrastructure that supports LDAPS
- An Active Directory global security group for administrators
- An Active Directory global security group for standard users (viewers)

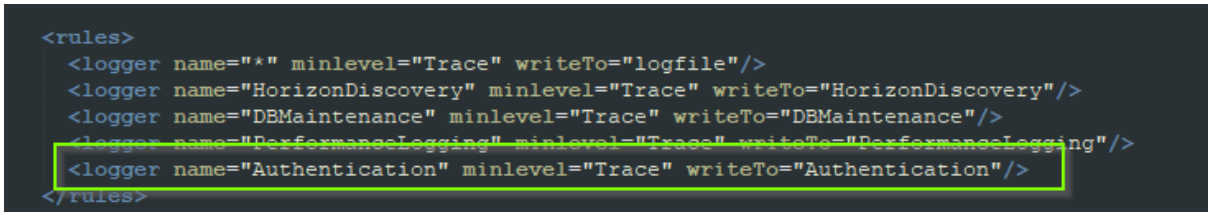
Note, most of this is case sensitive and really delicate / fussy. At the time of writing this, I'm having to rely on a novell .Net Core library which is very temperamental.

A few suggestions:

- Treat everything like its case sensitive.
- Check the object properties in Active Directory and copy / paste into the config file.
- Restart the service between each modification.
- Remember that the base search path must contain the users along with the group memberships you care about.

Extended Logging:

Modify the nlog.config file and change the below line as follows:



For what it's worth, I apologise this is such a pain, it was a pain just to get working.