How Things Work #1

# Parallel Installation

## Introduction

The actual installation portion of provisioning servers is perhaps the most time consuming task. Copying files to the server, unarchiving them, and performing the installation tasks are all disk-intensive, and somewhat network intensive. That means it can take a long time to deploy multiple servers – either because you wait for each to finish before starting the next, or because they are contending for network bandwidth.

The extreme version of this problem occurs while handling large cluster installation automation, where dozens or hundreds of machines are being spun up, and all must complete to make the cluster fully functional.

Stacki resolves this issue by utilizing a massively parallelized installer that can take advantage of an increasing number of machines to install entire clusters in parallel. Stacki uses a variant of BitTorrent to turn your installation network into a peer-to-peer RPM sharing network. We're going to focus on how Avalanche is utilized inside the Stacki installer, and why it's important for the average organization to get these benefits.

We're going to focus on how the installer is utilized inside the Stacki installer, and why it's important for the average organization to get these benefits. Note that this information also applies generally to the StackIQ Boss commercial product.

## What Stacki Does to Parallelize Installs

In your standard "golden image" deployment tool, while the image is being spawned out to server number one, the network connection on the headend is busy servicing that download. While more than one image can be downloaded to backend servers at a time, each one that is added slows the delivery of all images, just as increasing the number of active users decreases the response time of a website. The installer avoids this problem through several methods that Stacki takes advantage of.

## Background Information

Stacki is an RPM-based installer. This means that the base image is in an installation ISO from the Linux vendor (RedHat, CentOS, Oracle, Scientific, or other RedHat derivatives), and all of the additional software packages are in RPMS. This is important, because RPMS are both segmented chunks for transferring across a network (a single RPM is much smaller than a golden image, even if the sum of all RPMS is much bigger). It is also important because RPMS are more adapt-

able. They are designed to be installed on a variety of systems, and require only a valid Linux image that they target.
The other point to be aware of for this discussion is the private network in a Stacki environment. The private network connects every single server Stacki manages. While it is normally discussed in terms of connecting them to the headend, so Stacki can do installs and reinstalls, the truth that is important for this overview is that it interconnects all of them.

## The Communications

The first thing Stacki does on an install is send down a small starter OS that gets installed into a RAMdisk on the target machine. This starter install is the tool that then performs the actual installation. By using this method, things like RAID configuration are possible, and the installation can assume nothing about the disks and configure the machine as needed.

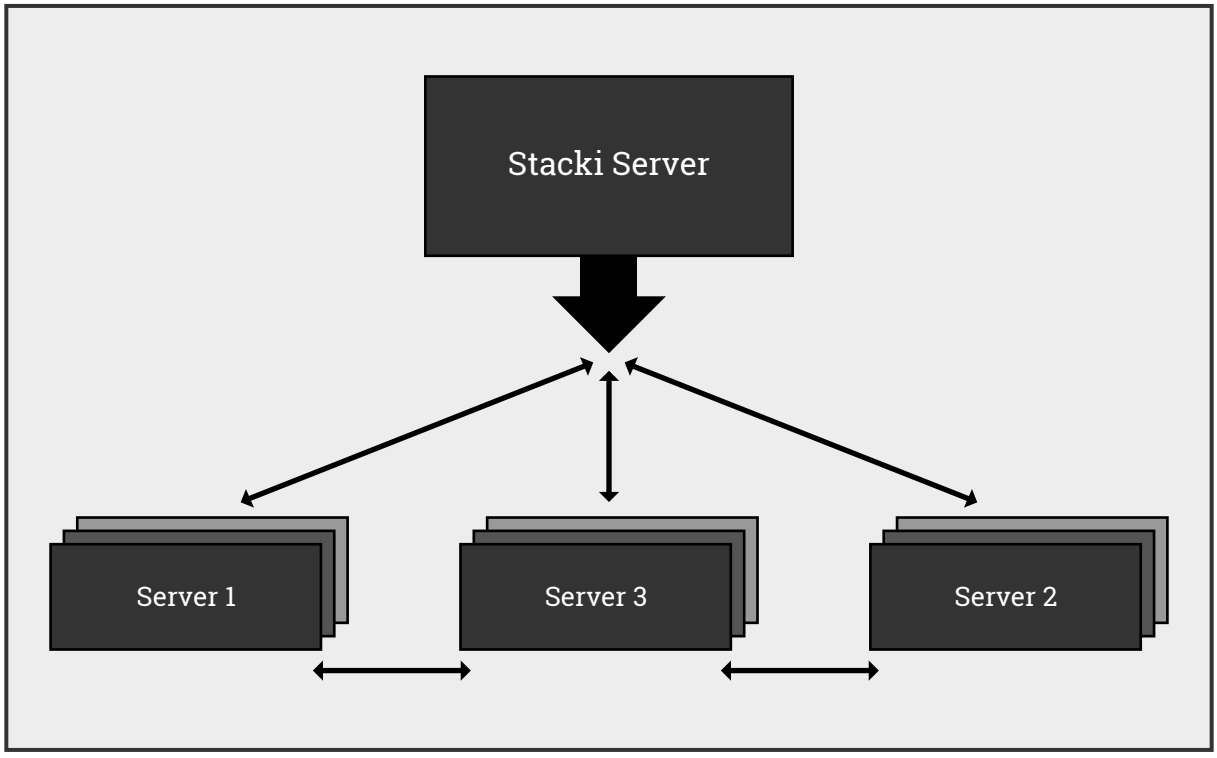
This small pre-install boot includes the installer.  This installer then utilizes Kickstart to begin pulling the RPMs necessary for the installation from the frontend. Since this initial RAMDisk boot is small, it does not monopolize the headend's network connection in the manner that transferring golden images would.

## Where the Installer Kicks in

This is the point at which the installer begins offering parallelization benefits.  As files are sent to backend servers, they update the availability of those files, so that each server becomes a mini-RPM server. That means the headend isn't shouldering all of the installation burden as it would on a golden image or less sophisticated installation tool. Instead, each server is capable of serving up the RPMS it already has. That makes the installation faster as machines share the burden.

In short, the overall network becomes the bottleneck (assuming you can fill it with your installations), not a single NIC on a single server.

This is what the installation communication looks like:



The Stacki server will serve up the RPM to a backend server – say server 1, though it doesn't have to be – and after that, when requesting a file, either the Stacki server or server 1 can respond with "I have it, here".

Experience shows that the more servers added, the more time this saves. As a rule of thumb for Stacki, "Two or Twenty, the time is about the same." When talking to people about StackIQ Boss – the genetic parent of Stacki, the equivalent statement would be "20 or 200, the time is about the same." That is because Boss is aimed at larger clustered environments. The point is, the installer is fast. Very fast.

## Summary

Stacki uses a number of underlying tools to achieve its goals of fast, reliable, and repeatable installs. Some of those technologies are pretty advanced – like the installer – and all of them work together under the Stacki hood to help you automate your data center.