# Plan

## Create host host capable of routing commands to remote services

* Don’t worry about re-try logic
* End-to-end test with two running remote instances would be nice
* Need to add some integration tests that cut out dbus and the remote features – that way more logic can be tested on travis

## Create a sample application that makes use sending commands

* Start with 2 services
* Maybe have services written in 2 different languages
* Sample application forces me to write clients early which should make integration with host easier

## Add event publishing and subscription

## Add use of events in sample application

## Fault tolerance - Implement error queues, re-try logic

* Be careful with events – make sure test-cases include situations where events reach one subscriber but not other(s)

## Update sample application to demonstrate fault-tolerance features

## Start to add monitoring capabilities

* Have a look at NServiceBus and other examples

## Update sample application to use monitoring capabilities

## Do some research and add capabilities for Sagas or workflows

## Make use of Saga/workflow features in sample application

## Checklist / not sure where to put them

1. Can I capture user intent and make it safer even faster?
2. Invalid configurations are detected and handled accordingly
   1. e.g. 2 handlers registered for same command
3. Refactor multi-jvm tests to be more expressive? e.g. createHostName “host1” inCluster “cluster” withConfigs host1Configs
4. Is it easy for people to use the project
   1. have all dependencies been mentioned and shown how to set up?
5. Joining a cluster dynamically
   1. a new node can be added to the system and can join an existing cluster
      1. will need to get and share configs
6. Configuration validation protocol
   1. nodes in the cluster will regularly check that their configs are all identical
7. Even if a physical machine dies, messages that are in a queue are not lost
8. The decision to use one akkesb host per-application or multiple applications using a single host has been carefully considered and the reasons for selection are still valid.
9. Performance – can message routing performance be improving by using different routing strategies or dispatchers?
10. Fault-tolerance / reliability – have a visual display of all messages being sent on the dead letters actor channel – check them regularly.
11. Akka's clustering features allow the load to be spread all around the available hardware – could there be some benefit to utilising this – rather than having one bus host application per-machine?
    1. Managing state might be a problem – but why? Distributed address book doesn't seem so bad?
12. Can I make the application more verbose?
    1. Logging certain events – important messages received or about to be sent
13. Have I applied risk management?
    1. “What if this goes wrong?”
    2. “What if this hardware/software component stops working?”
    3. “What if a queue goes down?”