> How will the parts of this system communicate those messages? Please produce some sort of diagram,  topology overview, or text describing this.

Each part of the system will have a selection of points, like a checklist, that must be met by the message that was received for the ready status code to be sent. Otherwise, a not ready status code will be sent. The checklist for certain parts of the system will most likely require ready status for other parts to be considered ready or useful. This will also allow smaller messages or tasks to be sent freely without worrying about the system being overloaded with larger tasks that cannot even be completed yet.

> How would you go about deploying this platform in a public cloud of your choice (describe your methodology and choice of tooling in 1+ paragraphs).

I would deploy this on a public cloud because the software will need to access lots of data from many different sources, so a public cloud is the most cost effective and simple way to not run into issues. I would also use Microsoft Azure, because it is more developed and more cost effective for large software that will be more consistently used. Additionally, many large companies already utilize Azure.

> How would you go about chunking this massive software into phases (what Minimally Viable Products would you create?). (1+ paragraphs describing at least 2 MVPs).

The first MVP I would create would simply be a system that can send and receive messages, and confirm that the message was received. Then I would create a second MVP that would have messages that require the confirmation of other received messages for them to be sent to begin with.

> How do you divide up tasks that can be delegated to other developers on the team should they be available (1+ paragraphs describing a simple project plan)?

I would delegate someone else to create the second MVP, this would allow the second MVP to be ready as soon as possible. Then I would delegate someone whose primary responsibility in this project would be scaling, so making sure that as the processes and software we create is expanded to a larger scale it still works and can be used efficiently. Then, someone else would be in charge of documentation, since there will be multiple people working on the project, there needs to be a way for everyone to understand what everyone else is doing and how that changes what they are doing; this person ideally would also sort out any bugs that come with slight changes from one persons work that might cause bigger problems in someone else's work.