How much resource to allocate for cleaning up tech debt?

Tech debt is accumulated through time - new features need to kick in as fast as possible, but no resource for refactoring or cleaning up, so developers have to use hacks ways to squeeze the new features into existing codes. If it works, it stays there.

Till some day you find that the tech debt is so big that you cannot just circumstance it - how do you squeeze new bricks into a Lego Empire Building that has too few logical and wholesome design, but has been built by squeezing new bricks into any space?

This is often the pivoting point where many teams need to take a call on: to pause for cleaning up tech debt, or push harder to carry on with the tech debt to not to slow down any new progress. When there's an external pressure, no matter from macro economy, competition in the same field, or from upper management, in many occasions people will choose the latter - that, just keep building new things, and forget about the tech debt. New things generate new impact, while cleaning up tech debts only leave a blank period that you cannot describe as "multiply metric A by B%".

However, tech debt won't just disappear like a magic. If continuing growing, it will eventually eliminate space for any new features; furthermore, it can even make debt cleaning itself too hard to do - how can you fix certain bricks of your fragile Lego Empire Building with 100% confidence it won't crash? You can't. And that's when people abandon the project.

You need to allocate resource(time, people) for cleaning up tech debt, or even better, set up good rules to limit the generation of tech debt.